การประยุกต์ใช้แนวคิดเชิงวัตถุในการออกแบบวงจรตรรกะเชิงผสม

นาย ศรัณย์ ชัยวรวิทย์กุล

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2546

ISBN 974-17-4026-3

APPLYING OBJECT ORIENTED CONCEPT TO COMBINATION LOGIC DESIGN

Mr. Sarun Chaiworawitgul

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Engineering in Computer Engineering

Department of Computer Engineering
Faculty of Engineering
Chulalongkorn University

Academic Year 2003

ISBN 974-17-4026-3

หัวข้อวิทยานิพนธ์	การประยุกต์ใช้แนวคิดเชิงวัตถุในการออกแบบวงจรตรรกะเชิงผสม
โดย	นายศรัณย์ ชัยวรวิทย์กุล
สาขาวิชา	วิศวกรรมคอมพิวเตอร์
อาจารย์ที่ปรึกษา	ผู้ช่วยศาสตราจารย์ ดร.พรศิริ หมื่นไชยศรี
อาจารย์ที่ปรึกษาร่วม	อาจารย์ ดร.โปรดปราน พิตรสาธร

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

..... คณบดี คณะวิศวกรรมศาสตร์
(ศาสตราจารย์ ดร.ดิเรก ลาวัณย์ศิริ)

คณะกรรมการสอบวิทยานิพนธ์

..... ประธานกรรมการสอบ
(ผู้ช่วยศาสตราจารย์ บุญชัย โสวรรณวณิชกุล)

..... อาจารย์ที่ปรึกษา
(ผู้ช่วยศาสตราจารย์ ดร.พรศิริ หมื่นไชยศรี)

.....อาจารย์ที่ปรึกษาร่วม
(อาจารย์ ดร.โปรดปราน พิตรสาธร)

.....กรรมการ
(อาจารย์ สมโชค เรืองอิทธินันท์)

ศรัณย์ ชัยวรวิทย์กุล : การประยุกต์ใช้แนวคิดเชิงวัตถุในการออกแบบวงจรตรรกะเชิงผสม. (APPLYING OBJECT ORIENTED CONCEPT TO COMBINATION LOGIC DESIGN) อ.ที่ปรึกษา : ผู้ช่วยศาสตราจารย์ ดร. พรศิริ หมื่นไชยศรี, อ.ที่ปรึกษาร่วม : อาจารย์ ดร. โปรดปราน พิตรสาธร, 108 หน้า. ISBN 974-17-4026-3.

แนวคิดเชิงวัตถุได้รับความนิยมและประสบความสำเร็จอย่างมากสำหรับขั้นตอนของการออกแบบซอฟต์แวร์ ซึ่งการออกแบบฮาร์ดแวร์มีความยากและซับซ้อนมากขึ้นประกอบกับความต้องการลดเวลาและแรงงานในขั้นตอนการออกแบบ ดังนั้นวิทยานิพนธ์นี้จึงนำแนวคิดเชิงวัตถุมาออกแบบและสร้างเครื่องมือออกแบบวงจรตรรกะเชิงผสมเพื่อใช้ในการออกแบบฮาร์ดแวร์

แนวคิดเชิงวัตถุที่นำมาใช้ในการออกแบบเครื่องมือประกอบด้วย แนวคิดการห่อหุ้ม แนวคิดการถ่ายทอด และแนวคิดการนำกลับมาใช้ โดยเครื่องมือออกแบบวงจรตรรกะเชิงผสมแบบแนวคิดเชิงวัตถุที่ได้ออกแบบและสร้างขึ้นในวิทยานิพนธ์นี้ประกอบด้วย ส่วนออกแบบวงจรแบบกราฟิกซึ่งนำแนวคิดทั้งสามมาใช้ในการออกแบบ เครื่องมือสังเคราะห์วงจร และเครื่องมือจำลองการทำงาน

ผลลัพธ์จากเครื่องมือที่ได้ออกแบบและสร้างขึ้น เมื่อนำไปตรวจสอบความถูกต้องกับผลลัพธ์ของวงจรมาตรฐาน International Symposium Circuit and Systems 1985 (ISCAS85) ปรากฏว่า ผลลัพธ์ที่ได้มีความถูกต้องและตรงกัน ดังนั้นจึงสรุปได้ว่าวิทยานิพนธ์นี้ได้สร้างเครื่องมือใหม่สำหรับผู้ออกแบบฮาร์ดแวร์โดยใช้หลักแนวคิดเชิงวัตถุซึ่งสามารถทำงานได้อย่างถูกต้อง เพื่อให้การออกแบบฮาร์ดแวร์ได้รับความสะดวกและแม่นยำขึ้น และเป็นการสนับสนุนการเชื่อมโยงกันทางด้านความรู้และกระบวนการระหว่างซอฟต์แวร์กับฮาร์ดแวร์

ภาควิชา	วิศวกรรมคอมพิวเตอร์	ลายมือชื่อนิสิต_____
สาขาวิชา	วิศวกรรมคอมพิวเตอร์	ลายมือชื่ออาจารย์ที่ปรึกษา _____
ปีการศึกษา	2546	ลายมือชื่ออาจารย์ที่ปรึกษาร่วม _____

4470557221 : MAJOR COMPUTER ENGINEERING

KEY WORD: OBJECT ORIENTED CONCEPT / INHERITANCE / REUSABILITY / COMBINATION LOGIC DESIGN

SARUN CHAIWORAWITGUL : APPLYING OBJECT ORIENTED CONCEPT TO COMBINATION LOGIC DESIGN. THESIS ADVISOR : ASSISTANT PROFESSOR PORNSIRI MUENCHAISRI, Ph.D., THESIS COADVISOR : PROADPRAN PITSATORN, Ph.D., 108 pp. ISBN 974-17-4026-3.

While object-oriented technology is being adopted widely in software engineering, it is not much explored in the area of hardware design. The libraries that must be used directly and cannot be modified, or while the little function or property are changed mostly causes redesign of a circuit. Those are the problems that all hardware designers are facing currently.

This thesis proposes a new tool to help hardware designers in the process of hardware design, specifically to the combination logic design. This tool acquires the properties of object-oriented (OO) concept namely encapsulation, inheritance, and reusability. The proposed tool encompasses a hardware design drawing, a synthesizer, and a simulator with a friendly graphical user interface.

The output of the system (the combination logic circuit) is verified by comparing the result with the standard circuit by International Symposium Circuit and Systems 1985 (ISCAS85). As a result, This research offers a new tool for hardware designers that ease the process of hardware design. Finally, this thesis certainly promotes the complement of utilizing the useful concept of software in the world of hardware.

Department	Computer Engineering	Student's signature _____
Field of study	Computer Engineering	Advisor's signature _____
Academic year	2003	Co-advisor's signature _____

กิตติกรรมประกาศ

ขอขอบพระคุณ ผู้ช่วยศาสตราจารย์ ดร.พรศิริ หมื่นไชยศรี อาจารย์ ดร.โปรดปราน พิตรสาธร และผู้ช่วยศาสตราจารย์บุญชัย โสวรรณวณิชกุล ที่กรุณาให้คำปรึกษาและความช่วยเหลือในด้านต่างๆ ตลอดระยะเวลาที่ทำวิทยานิพนธ์ชิ้นนี้

ขอขอบพระคุณ อาจารย์สมโชค เรืองอิทธินันท์ ที่กรุณาให้คำแนะนำอันมีค่า ซึ่งช่วยให้วิทยานิพนธ์ชิ้นนี้มีความสมบูรณ์ในเนื้อหามากยิ่งขึ้น

ขอขอบพระคุณ นายพิศูจน์ ชัยวรวิทย์กุล นางกฤติยา ชัยวรวิทย์กุล นายประพจน์ ชัยวรวิทย์กุล นายศักดา ชัยวรวิทย์กุล นายกนกพล ชัยวรวิทย์กุล และนางสาวอทิตา ตันติวงษ์ ที่ได้ให้ความช่วยเหลือและการสนับสนุนในหลายๆ ด้าน ไม่ว่าจะเป็นวิชาการและกำลังใจ

ขอขอบพระคุณนายชนินทร์ มหารักษ์ นายชยันต์ เทพบุตร นายชาติชาย ดวงสอาด นายดนัย สุขจินดาเสถียร นายต้นพงศ์ วรรณะธูป นายนัทที นิภานันท์ นายประพนธ์ บวรภราดา นายปิยะ วราบุญทวีสุข นายรังสรรค์ เกียรติภานนท์ นางสาวเรียวไผ่ บุญเกิด และนายสมศักดิ์ ภัทรศุกล ที่ได้ให้คำแนะนำและช่วยอธิบายทฤษฎีและเทคนิคต่างๆ ที่ใช้ในการเขียนโปรแกรม

ศรัณย์ ชัยวรวิทย์กุล

17 มกราคม 2547

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

หน้า

สารบัญรูป

หน้า

สารบัญรูป (ต่อ)

หน้า

สารบัญรูป (ต่อ)

หน้า

สารบัญรูป (ต่อ)

หน้า

สารบัญรูป (ต่อ)

หน้า

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

การออกแบบฮาร์ดแวร์ในปัจจุบันมีความยุ่งยากและซับซ้อนมากกว่าในอดีต เนื่องจากการพัฒนาเทคโนโลยีเป็นไปอย่างรวดเร็ว ทำให้มีความต้องการนำด้านฮาร์ดแวร์ไปประยุกต์ใช้กับเทคโนโลยีในศาสตร์ด้านอื่นๆ เพิ่มมากยิ่งขึ้น เมื่อเทคโนโลยีมีความซับซ้อนมากขึ้นย่อมส่งผลให้การออกแบบฮาร์ดแวร์มีความซับซ้อนมากขึ้นตามลำดับ ความซับซ้อนของฮาร์ดแวร์ประกอบด้วยการทำงานของอุปกรณ์ภายในที่มีจำนวนมาก ที่ต้องสามารถทำงานร่วมกันและให้ผลลัพธ์ที่ถูกต้อง โดยที่อุปกรณ์ภายในแต่ละชนิดส่วนใหญ่แล้วจะมีลักษณะเป็นชิป (Chip) ซึ่งชิปหนึ่งตัว ประกอบไปด้วยเกท (Gate) ซึ่งเกิดจากการนำเอาทรานซิสเตอร์มาต่อเป็นวงจรไฟฟ้าเพื่อควบคุมการทำงานให้เป็นไปตามลอจิกชนิดต่างๆ เป็นจำนวนมากมายมหาศาล ปัจจัยสำคัญที่ถูกจำกัดต่อการออกแบบฮาร์ดแวร์ในปัจจุบันก็คือ เวลา เพราะผลิตภัณฑ์แต่ละชนิดในปัจจุบันมีวงจรชีวิตที่สั้นมาก อันเนื่องจากการเปลี่ยนแปลงเทคโนโลยีอย่างรวดเร็ว ดังนั้นนักออกแบบฮาร์ดแวร์จึงจำเป็นต้องใช้เวลาในการออกแบบที่ไม่ยาวนานจนเกินไป

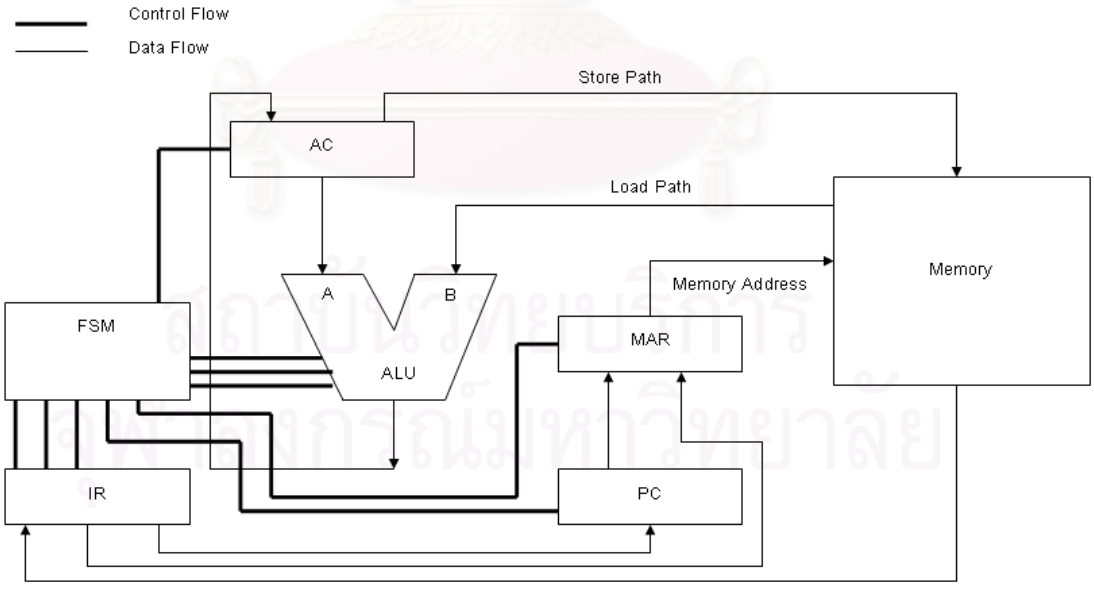
การออกแบบฮาร์ดแวร์เป็นเรื่องที่ละเอียดอ่อนมาก เนื่องจากจะต้องออกแบบจากสิ่งที่คลุมเครือและมีรายละเอียดที่ไม่มากเพื่อให้ได้มาซึ่งฮาร์ดแวร์ที่ทำงานได้ถูกต้องและชัดเจน ประกอบกับเครื่องมือหรือเทคโนโลยีที่ใช้ในการออกแบบมีมากมายและเกิดขึ้นมาอย่างรวดเร็ว หรือมีการเปลี่ยนแปลงปรับปรุงพัฒนาให้ดีขึ้น เช่น งานวิจัยของ Brad L. และ Brent E. [13] ที่ได้ศึกษาวิจัยเกี่ยวกับการนำเอาโปรแกรมภาษาจาวา (Java Programming Language) มาสร้างเครื่องมือที่ใช้ในการออกแบบฮาร์ดแวร์หรือที่เรียกว่า “เจเอชดีแอล” (Java Hardware Description Language : JHDL)

ถึงแม้ว่าเครื่องมือที่ใช้ในการออกแบบฮาร์ดแวร์จะมีการพัฒนาหรือปรับปรุงให้ดีขึ้น แต่อย่างไรก็ตามพื้นฐานของการออกแบบฮาร์ดแวร์ก็ยังคงเป็นวิธีดั้งเดิม ซึ่งมีพื้นฐานมาจากการใช้ทฤษฎีของพีชคณิตแบบบูลีน (Boolean) ช่วยในการออกแบบ และมีการใช้เกทเป็นสัญลักษณ์ในสมการบูลีน การออกแบบฮาร์ดแวร์ในปัจจุบันมีความยากและความซับซ้อนมาก หากเกิดความผิดพลาดจากการออกแบบจะทำให้สูญเสียเวลาในการศึกษาและแก้ไขการออกแบบฮาร์ดแวร์ที่ได้ออกแบบไว้แล้วไปอีกมาก จากปัญหาของการออกแบบฮาร์ดแวร์ข้างต้นจึงควรมีการศึกษาแสวงหาแนวทางสำหรับการออกแบบฮาร์ดแวร์ด้วยวิธีใหม่เพื่อช่วยลดระยะเวลาในการออกแบบฮาร์ดแวร์

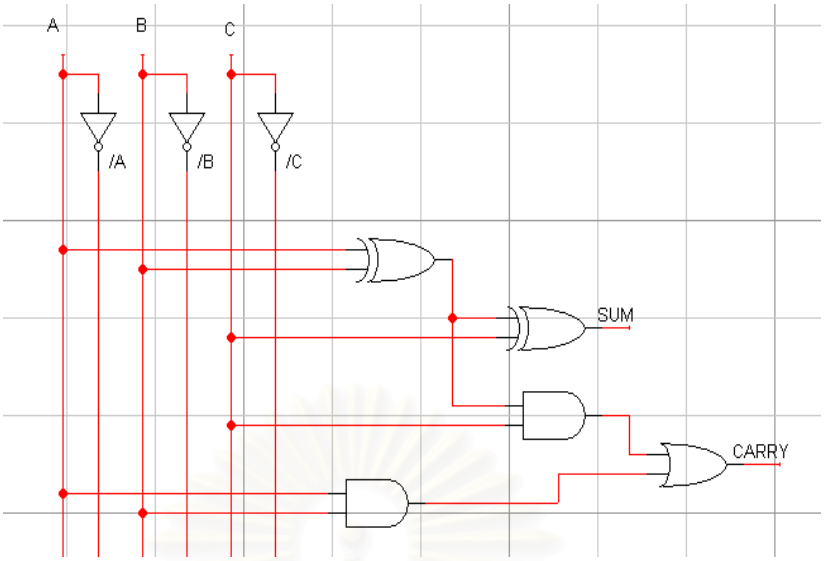
เทคโนโลยีทางด้านการออกแบบด้วยแนวคิดเชิงวัตถุเป็นแนวทางที่ได้รับความนิยมเป็นอย่างมากในมุมมองด้านซอฟต์แวร์ ด้วยคุณสมบัติที่สอดคล้องกับโลกของวัตถุซึ่งเปรียบเทียบได้กับ

โลกของความเป็นจริง ในวงการซอฟต์แวร์ได้นำเอาวิธีการออกแบบด้วยแนวคิดเชิงวัตถุมาใช้ในการแก้ปัญหาทางด้านการออกแบบ ซึ่งปรากฏว่าประสบความสำเร็จเป็นอย่างสูง กระบวนการทางด้านการออกแบบด้วยแนวคิดเชิงวัตถุจึงได้รับความสนใจจากผู้ออกแบบทางด้านฮาร์ดแวร์เป็นจำนวนมาก ด้วยความคิดว่า การออกแบบด้วยแนวคิดเชิงวัตถุจะสามารถจัดการกับปัญหาความซับซ้อนในการออกแบบฮาร์ดแวร์ได้เป็นอย่างดี และช่วยตรวจสอบความถูกต้องในการออกแบบให้ดียิ่งขึ้น ดังนั้นนักออกแบบฮาร์ดแวร์บางกลุ่มได้พยายามนำแนวคิดเชิงวัตถุมาประยุกต์ใช้กับการออกแบบทางด้านฮาร์ดแวร์ [9]

การออกแบบฮาร์ดแวร์เป็นกระบวนการที่ซับซ้อน โดยที่อุปกรณ์หนึ่งๆ ประกอบด้วยชิ้นส่วนหรืออุปกรณ์ต่างๆ มากมาย เช่น การออกแบบซีพียูเบื้องต้น [19] ดังรูปที่ 1.1 ซีพียูเบื้องต้นประกอบด้วยส่วนประกอบต่างๆ เช่น หน่วยคำนวณและตรรกะ (Arithmetic Logic Unit : ALU) หน่วยความจำ (Memory) และเรจิสเตอร์ (Register) ชนิดต่างๆ เช่น Memory Address Register (MAR) สำหรับหน่วยคำนวณและตรรกะซึ่งใช้ในการคำนวณค่าทางคณิตศาสตร์ ซึ่งหน่วยคำนวณและตรรกะแต่ละตัวจะทำหน้าที่บางอย่างคล้ายกันโดยมีสิ่งที่แตกต่าง คือ จำนวนบิตที่ใช้ในการคำนวณ หรือหน้าที่ในการคำนวณ ดังนั้นการนำคุณสมบัติของแนวคิดเชิงวัตถุมาประยุกต์ใช้ในการออกแบบฮาร์ดแวร์ คาดว่าจะสามารถลดเวลาและแรงงานในการออกแบบฮาร์ดแวร์ได้ ในหน่วยคำนวณและตรรกะอาจจะประกอบไปด้วยวงจรบวก (Adder) มีหน้าที่สำหรับบวกเลข วงจรบวกซึ่งเป็นวงจรตรรกะเชิงผสมจะประกอบไปด้วยเกทชนิดต่างๆ ดังรูปที่ 1.2



รูปที่ 1.1 การออกแบบซีพียูเบื้องต้น



รูปที่ 1.2 วงจรเต็มบวก (Full Adder) ขนาดหนึ่งบิต

รูปที่ 1.2 แสดงถึงวงจรเต็มบวกขนาดหนึ่งบิต ซึ่งประกอบไปด้วยเกทชนิดต่างๆ คือ แอนด์ (AND) ออร์ (OR) ออร์เฉพาะ (XOR) และตัวผกผัน (NOT) โดยที่เกทแต่ละชนิดมีหน้าที่การทำงานเป็นลักษณะเฉพาะ เช่น แอนด์เกท ทำหน้าที่ คำนวณค่าพีชคณิตบูลีนของแอนด์ ซึ่งแอนด์เกททุกตัวจะมีพฤติกรรมของการทำงานลักษณะเดียวกัน แต่สิ่งที่อาจแตกต่างกันในแอนด์เกทแต่ละตัว คือ จำนวนอินพุต ดังนั้นจะเห็นได้ว่าน่าจะสามารถนำแนวคิดการถ่ายทอดของแนวคิดเชิงวัตถุสามารถมาประยุกต์ใช้ในการออกแบบลักษณะนี้ได้เพื่อให้การออกแบบง่ายยิ่งขึ้น นอกเหนือจากแนวคิดการถ่ายทอดแล้ว ยังมีแนวคิดอีกสองประการที่มีความเชื่อมโยงกับงานวิจัยนี้ คือ แนวคิดการห่อหุ้ม (Encapsulation) และแนวคิดการนำกลับมาใช้ (Reusability) ซึ่งก่อให้เกิดประโยชน์ในงานการออกแบบฮาร์ดแวร์ โดยใช้แนวคิดเชิงวัตถุทั้งสามนี้สำหรับออกแบบวงจรตรรกะเชิงผสมในระดับเกท เนื่องจากเกทเป็นระดับพื้นฐานเบื้องต้นของการออกแบบฮาร์ดแวร์ ยังไม่พบงานวิจัยใดเลยที่นำแนวคิดเชิงวัตถุมาประยุกต์ใช้ในการออกแบบฮาร์ดแวร์ระดับเกท ดังนั้นงานวิจัยนี้น่าจะนับได้ว่าเป็นงานริเริ่มและเป็นงานที่เชื่อมโยงระหว่างโลกของฮาร์ดแวร์และซอฟต์แวร์สมัยใหม่เข้ามาด้วยกันในการออกแบบฮาร์ดแวร์ด้วยแนวคิดเชิงวัตถุ

1.2 วัตถุประสงค์ของการวิจัย

งานวิจัยนี้มีวัตถุประสงค์สร้างเครื่องมือสำหรับออกแบบวงจรตรรกะเชิงผสมโดยนำแนวคิดการห่อหุ้ม (Encapsulation) แนวคิดการถ่ายทอด (Inheritance) และแนวคิดการนำกลับมาใช้ (Reusability) ของแนวคิดเชิงวัตถุมาใช้สำหรับการออกแบบวงจรตรรกะเชิงผสม เพื่อช่วยลดความซับซ้อน เวลา และช่วยตรวจสอบความถูกต้องในขั้นตอนการออกแบบฮาร์ดแวร์สำหรับผู้ออกแบบฮาร์ดแวร์

1.3 ขอบเขตของการวิจัย

1. สามารถใช้แนวคิดการห่อหุ้ม การถ่ายทอด และการนำกลับมาใช้ของแนวคิดเชิงวัตถุมาใช้ในการออกแบบวงจรตรรกะเชิงผสมในระดับเกทได้
2. ความถูกต้องของการออกแบบวงจรตรรกะเชิงผสมแบบแนวคิดเชิงวัตถุ จะกระทำโดยเปรียบเทียบค่าความจริงของผลลัพธ์จากวงจรมาตรฐาน ISCAS85 จำนวนหกวงจร
3. เครื่องมือออกแบบวงจรตรรกะเชิงผสมแบบแนวคิดเชิงวัตถุที่สร้างขึ้นในวิทยานิพนธ์นี้จะไม่คำนึงถึงดีเลย์ในขั้นตอนการจำลองการทำงาน

1.4 ประโยชน์ที่ได้รับ

1. ได้มาซึ่งเครื่องมือสำหรับให้ผู้ออกแบบฮาร์ดแวร์ใช้ในการออกแบบฮาร์ดแวร์ โดยลดเวลาในการออกแบบฮาร์ดแวร์
2. เป็นต้นแบบและแนวทางในการนำแนวคิดเชิงวัตถุมาใช้ในการออกแบบฮาร์ดแวร์เบื้องต้น
3. เป็นต้นแบบและแนวทางในการลดความซับซ้อนและลดเวลาในขั้นตอนการออกแบบฮาร์ดแวร์

บทที่ 2
เอกสารและงานวิจัยที่เกี่ยวข้อง

2.1 ทฤษฎีที่เกี่ยวข้อง

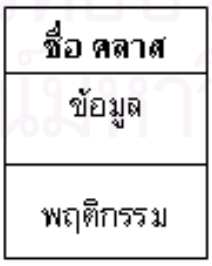
ทฤษฎีที่มีความสำคัญสำหรับงานวิจัยนี้ประกอบด้วยสองทฤษฎีหลัก ได้แก่ ทฤษฎีแนวคิดเชิงวัตถุ และทฤษฎีลอจิกเกท

2.1.1 แนวคิดเชิงวัตถุ (Object-Oriented Concept)

แนวคิดเชิงวัตถุประกอบด้วยคุณสมบัติที่สำคัญหลายลักษณะ ซึ่งทำให้แนวคิดเชิงวัตถุ ได้รับความนิยมเป็นอย่างสูง คุณสมบัติที่สำคัญของแนวคิดเชิงวัตถุ มีอยู่สี่ลักษณะ คือ 1) คุณสมบัติการห่อหุ้ม 2) คุณสมบัติการถ่ายทอด 3) คุณสมบัติการนำกลับไปใช้ และ 4) คุณสมบัติโพลิมอร์ฟีซึม [16] สำหรับวิทยานิพนธ์นี้ได้นำแนวคิดเชิงวัตถุมาสร้างเครื่องมือออกแบบวงจรตรรกะเชิงผสม โดยคุณสมบัติของแนวคิดเชิงวัตถุที่นำมาใช้ในการสร้างเครื่องมือดังกล่าวประกอบด้วยสามคุณสมบัติ คือ 1) คุณสมบัติการห่อหุ้ม 2) คุณสมบัติการถ่ายทอด และ 3) คุณสมบัติการนำกลับมาใช้ เนื่องจากเป็นคุณลักษณะที่เกี่ยวข้องกับงานออกแบบฮาร์ดแวร์

2.1.1.1 คุณสมบัติการห่อหุ้ม (Encapsulation) คือ การนำเอาข้อมูลมาผูกรวมไว้กับการกระทำที่เกี่ยวข้องกับข้อมูลให้เป็นโครงสร้างหนึ่งหน่วย [17] ซึ่งมีส่วนประกอบทั้งหมดสองส่วน คือ คลาสและวัตถุ (Classes and Objects) และ สาร (Messages)

คลาสและอ็อปเจกต์ คลาสในแนวคิดของภาษาเชิงวัตถุ (Object-oriented language) แสดงถึงเนื้อหาและพฤติกรรมของเอนทิตี (Entity) ในโลกของความจริง โดยห่อหุ้มข้อมูล (Attribute) และสาระสำคัญของพฤติกรรม (Operation, Method, Service) ที่เกี่ยวข้องดังรูปที่ 2.1

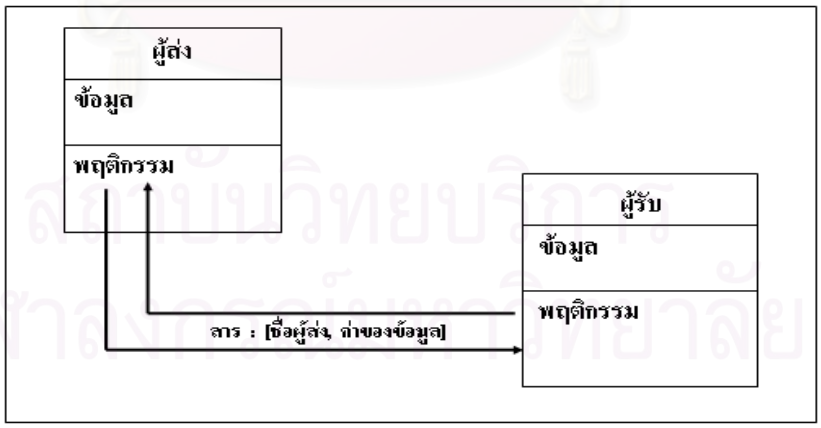


รูปที่ 2.1 โครงสร้างของคลาส

ในแนวคิดเชิงวัตถุจะมองว่าในชีวิตจริงนั้นทุกสิ่งทุกอย่างย่อมมีลักษณะหรือคุณสมบัติเป็นของตนเอง ซึ่งก็คือ ข้อมูล เช่น คนต้องมีข้อมูล

ของวันเกิด ชื่อ หรือสีผิว ซึ่งคุณสมบัติเหล่านี้คือความสัมพันธ์ระหว่างคลาสกับโดเมนของสิ่งที่เกี่ยวข้อง ซึ่งโดเมนที่เกี่ยวข้องคือ เซตของค่าที่กำหนด เช่น เมื่อกล่าวถึงคลาสของรถยนต์ที่มีข้อมูลเป็นสี จะได้โดเมนคือ {สีขาว สีส้ม สีเขียว ฯลฯ} นอกจากจะมีการกำหนดคุณลักษณะให้กับคลาสแล้ว ข้อมูลยังสามารถเก็บค่า (Value) ได้อีกด้วย เช่น สีของรถยนต์ซึ่งอาจจะเป็นสีใดๆ ก็ได้ที่อยู่ในโดเมน สำหรับพฤติกรรม คือ อัลกอริทึม (Algorithm) ที่ใช้ในการจัดการกับข้อมูลที่คลาสห่อหุ้มไว้ พฤติกรรมจะถูกห่อหุ้มและมีคลาสเป็นตัวบ่งบอกถึงพฤติกรรมนั้นๆ คลาสอาจถูกกำหนดให้เป็นรูปแบบที่ใช้บอกถึงลักษณะที่มีความคล้ายคลึงกันของวัตถุที่คล้ายคลึงกัน ดังนั้นวัตถุ คือ โครงสร้างที่ได้รับการถ่ายทอดข้อมูลและพฤติกรรมมาจากคลาสหนึ่งๆ โดยสามารถกล่าวอีกนัยหนึ่งได้ว่า ซูเปอร์คลาส (Superclass) เป็นที่เก็บรวบรวมคลาส และสับคลาส (Subclass) เป็นวัตถุของคลาส

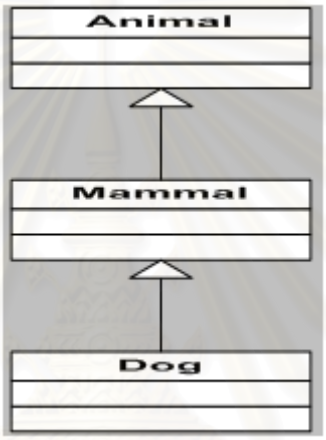
- สาร (Messages) คือ ปฏิสัมพันธ์ที่เกิดขึ้นระหว่างวัตถุ ซึ่งสารจะก่อให้เกิดการทำงานของวัตถุที่ถูกเรียกใช้งานโดยที่ผู้ส่ง (Sender) วัตถุจะสร้างสารจากพฤติกรรมที่มีอยู่ และสารยังช่วยในการผูกระบบเชิงวัตถุไว้ด้วยกัน เนื่องจากความสัมพันธ์ระหว่างพฤติกรรมของแต่ละวัตถุที่เป็นอิสระต่อกันเมื่อถูกนำมารวมกันจึงเป็นระบบเชิงวัตถุ ดังแสดงการทำงานของสารในรูปที่ 2.2



รูปที่ 2.2 การทำงานของสาร

2.1.1.2 คุณสมบัติการถ่ายทอด (Inheritance) คือ การที่ข้อมูลและพฤติกรรมที่มีความสัมพันธ์กับคลาสลูก (Child Class) ได้รับการขยาย (Extension) คุณสมบัติมาจากคลาสพ่อ (Parent class) โดยที่สับคลาสหรือคลาสลูก (Subclass หรือ Child

Class) จะได้รับคุณสมบัติทั้งหมดมาจากคลาสพ่อ นอกจากนี้แล้วสับคลาสยังอาจจะมีคุณสมบัติที่นอกเหนือไปจากคลาสพ่อได้อีกด้วย ซึ่งความสามารถในการขยายนี้ช่วยเพิ่มประสิทธิภาพของคุณสมบัติการถ่ายทอดให้สูงขึ้น นอกจากนี้แล้วการถ่ายทอดยังสามารถกระทำได้มากกว่าหนึ่งรุ่นอีกด้วย ดังแสดงตัวอย่างในรูปที่ 2.3 นอกจากสับคลาสจะรับคุณสมบัติมาจากคลาสพ่อโดยตรงแล้ว สับคลาสยังสามารถที่จะเปลี่ยนแปลงพฤติกรรมที่ได้รับมาจากซูเปอร์คลาสอีกด้วย ลักษณะการทำงานเช่นนี้เรียกว่า การโอเวอร์ไรด์ (Override) ซึ่งคือ การถ่ายทอดคุณสมบัติทางด้านข้อมูลและพฤติกรรม แต่มีการเปลี่ยนแปลงหรือแก้ไขคุณสมบัติบางอย่างเหล่านี้เพื่อให้มีความเหมาะสมกับคลาสใหม่



รูปที่ 2.3 ตัวอย่างของแนวคิดการถ่ายทอด

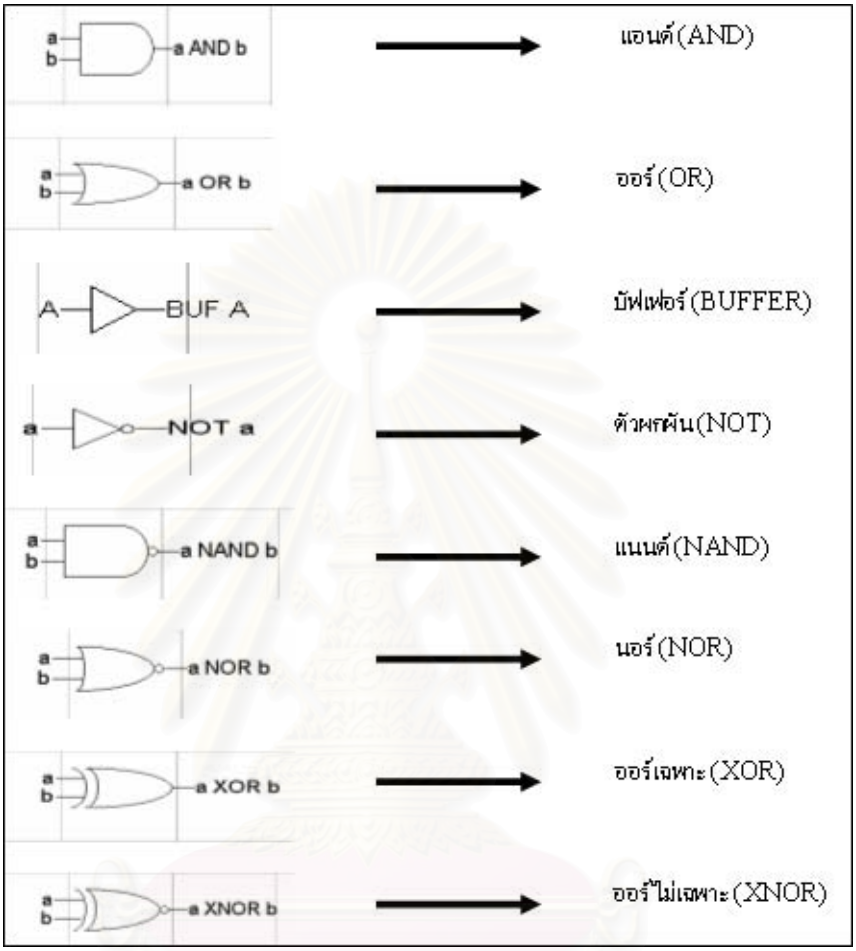
2.1.1.3 คุณสมบัติการนำกลับมาใช้ (Reusability) คือ คุณสมบัติที่ช่วยลดเวลาและเพิ่มความถูกต้องของการเขียนโปรแกรม โดยสิ่งที่ได้ออกแบบสำหรับการนำกลับมาใช้จะอยู่ในรูปแบบของไลบรารี (Library) วิธีการนำกลับมาใช้สามารถแบ่งออกได้เป็นสองวิธี คือ

1) การนำสิ่งที่ได้ออกแบบไว้สมบูรณ์แล้วซึ่งอยู่ในรูปแบบของไลบรารีมาใช้โดยตรงเพื่อเป็นส่วนหนึ่งของการออกแบบ
2) การนำสิ่งที่ได้ออกแบบไว้สมบูรณ์แล้วซึ่งอยู่ในรูปแบบของไลบรารีมาปรับปรุงเพื่อให้ตรงกับความต้องการในการออกแบบงานนั้นๆ

2.1.2 ลอจิกเกท (Logic Gates)

ลอจิกเกทเป็นที่นิยมอย่างมากในการใช้อธิบายถึงนิพจน์บูลีน (Boolean Expression) โดยที่แต่ละเครื่องหมายของลอจิก จะใช้สัญลักษณ์แทนด้วยเกทชนิดต่างๆ ซึ่งลอจิกเกทที่ได้นำมาพิจารณาในวิทยานิพนธ์นี้ ประกอบไปด้วยแปดลอจิกเกท ได้แก่ แอนด์ (AND) ออร์ (OR) บัฟเฟอร์

(BUFFER) ตัวผกผัน (NOT) แนนด์ (NAND) นอร์ (NOR) ออร์เฉพาะ (XOR) และออร์ไม่เฉพาะ (XNOR) ดังแสดงในรูปที่ 2.4



รูปที่ 2.4 สัญลักษณ์ของเกทแปดชนิด

2.2 งานวิจัยที่เกี่ยวข้อง

จากการค้นคว้างานวิจัยที่เกี่ยวข้องกับการนำวิธีการออกแบบด้วยแนวคิดเชิงวัตถุมาประยุกต์ใช้กับการออกแบบทางด้านฮาร์ดแวร์ พบว่ามีงานวิจัยลักษณะนี้อยู่สี่กลุ่มหลัก คือ

1) การนำคุณสมบัติการออกแบบเชิงวัตถุไปเพิ่มเติมเข้ากับภาษาเอชดีแอล (Hardware Description Language, HDL)
2) การนำแนวคิดเชิงวัตถุมาประยุกต์กับระบบที่ประกอบด้วยฮาร์ดแวร์และซอฟต์แวร์ (Object-Oriented System Engineering : OOSE)
3) การนำกระบวนการนำกลับมาใช้ของการออกแบบเชิงวัตถุมาออกแบบฮาร์ดแวร์
4) การสังเคราะห์วงจร (Synthesis) และการทวนสอบ (Verification)

โดยการใช้แนวคิดเชิงวัตถุ ซึ่งรายละเอียดของการนำวิธีการออกแบบด้วยแนวคิดเชิงวัตถุมาประยุกต์ใช้กับการออกแบบทางด้านฮาร์ดแวร์ มีดังนี้

2.2.1 การนำคุณสมบัติการออกแบบเชิงวัตถุไปเพิ่มเติมเข้ากับภาษาเอชดีแอล

งานวิจัยที่เกี่ยวข้องกับการนำคุณสมบัติของการออกแบบเชิงวัตถุไปเพิ่มเติมเข้ากับภาษาเอชดีแอล ซึ่งเป็นภาษาที่นิยมในการใช้สำหรับการออกแบบฮาร์ดแวร์ภาษาหนึ่งนอกเหนือจากภาษาที่นิยมอื่นๆ เช่น วีเอชดีแอล (Very high speed integrated circuit Hardware Description Language, VHDL) และ เวอริลอก (Verilog) ภาษาวีเอชดีแอลเป็นภาษาหนึ่งที่มีการนำคุณสมบัติเชิงวัตถุไปขยายเพิ่มเติมเข้ากับภาษาแทนที่จะสร้างภาษาขึ้นมาใหม่ เนื่องจากการสร้างภาษาขึ้นใหม่เป็นการเพิ่มเวลาและแรงงานรวมทั้งมีโอกาสเกิดข้อผิดพลาดได้มากกว่าการพัฒนาจากภาษาที่มีอยู่เดิม [6][14] ซึ่งงานวิจัยที่เกี่ยวข้องกับภาษาวีเอชดีแอลก็คือ การนำเอาคุณสมบัติเชิงวัตถุมาขยายเพิ่มเติมเข้ากับภาษาเดิมที่มีอยู่ เช่น งานวิจัยที่ศึกษาเกี่ยวกับการนำเอาคุณสมบัติการถ่ายทอด ซึ่งเป็นคุณสมบัติหนึ่งที่มีความสำคัญของแนวคิดเชิงวัตถุมาขยายเพิ่มเติมเข้ากับภาษาวีเอชดีแอลทางด้านสัญญาณ [1] หรืองานวิจัยที่ได้อธิบายถึงไวยกรณ์ (Syntax) ของคุณสมบัติเชิงวัตถุที่ขยายเพิ่มเติมเข้ากับภาษาวีเอชดีแอล [6] สำหรับภาษาเวอริลอกได้มีการนำคุณสมบัติของแนวคิดเชิงวัตถุ คือ คุณสมบัติการห่อหุ้ม ซึ่งจะรวบรวมข้อมูลกับพฤติกรรมไว้เป็นโครงสร้างหนึ่งหน่วยมาใช้งาน แต่ยังไม่พบการนำคุณสมบัติการถ่ายทอดและโพลีมอร์ฟิซึมมาประยุกต์ใช้ [23] และการนำภาษาวีเอชดีแอลมาเปรียบเทียบคุณสมบัติต่างๆ รวมทั้งแนวคิดเชิงวัตถุกับภาษาเวอริลอก [9][10]

2.2.2 การนำแนวคิดเชิงวัตถุมาประยุกต์กับระบบที่ประกอบด้วยฮาร์ดแวร์และซอฟต์แวร์

เนื่องจากระบบโดยส่วนใหญ่แล้วจะต้องมีซอฟต์แวร์ควบคุมการทำงานของฮาร์ดแวร์ แต่การออกแบบระบบมักจะแยกการออกแบบเป็นส่วน คือ ส่วนที่เป็นฮาร์ดแวร์และส่วนที่เป็นซอฟต์แวร์ ดังนั้นจึงมีงานวิจัยที่ศึกษาการออกแบบระบบโดยออกแบบทั้งฮาร์ดแวร์และซอฟต์แวร์ของระบบไปพร้อมกันเพื่อลดต้นทุนทางด้านเวลา โดยใช้กระบวนการเชิงวัตถุซึ่งถูกเรียกว่าวิศวกรรมระบบโดยใช้โมเดลเชิงวัตถุ (Model-based Object Oriented System Engineering : MOOSE) ดังงานวิจัยซึ่งนำเสนอแนวคิดการออกแบบโดยรวมซอฟต์แวร์และฮาร์ดแวร์เข้าด้วยกัน [2] งานวิจัย ซึ่งนำเสนอวิธีเชื่อมโยงระหว่างข้อกำหนดกับการออกแบบของระบบแบบเบ็ดเสร็จ [3] งานวิจัยที่นำเสนอการสร้างโมเดลของระบบซึ่งประกอบไปด้วยฮาร์ดแวร์และซอฟต์แวร์ [5] งานวิจัยซึ่งใช้ระเบียบแนวคิดเชิงวัตถุในการออกแบบระบบควบคุมฮาร์ดแวร์ [4] งานวิจัยซึ่งนำเสนอการใช้ภาษาวีเอชดีแอลในการพัฒนาวิศวกรรมระบบตามแบบแนวคิดเชิงวัตถุ [11] และงานวิจัยที่นำเสนอข้อกำหนดของบทบาทและหน้าที่การสร้างและใช้วัตถุในการออกแบบระบบ[12] ซึ่งงาน

วิจัยทั้งหมดที่ได้กล่าวมานี้จะเป็นการออกแบบระบบที่มีความเฉพาะเจาะจง (Specific Application)

2.2.3 การนำกระบวนการนำกลับมาใช้ของการออกแบบเชิงวัตถุมาออกแบบฮาร์ดแวร์

การนำแนวคิดการนำกลับมาใช้ของการออกแบบเชิงวัตถุมาประยุกต์สำหรับการออกแบบฮาร์ดแวร์เป็นอีกแนวทางหนึ่งที่มีการนำเอาประโยชน์ของแนวคิดเชิงวัตถุมาประยุกต์ใช้ เพื่อช่วยลดต้นทุนทางด้านเวลา แรงงานและข้อผิดพลาดจากการออกแบบ สำหรับงานวิจัยที่เกี่ยวข้องกับการนำกลับมาใช้ได้ปรากฏในรูปแบบของการออกแบบฮาร์ดแวร์เพื่อนำกลับมาใช้ และการนำฮาร์ดแวร์ที่ถูกสร้างไว้สำหรับการนำกลับมาใช้มาใช้งาน ดังเช่น งานวิจัย ซึ่งนำเสนอข้อกำหนดและวิธีการกำหนดฮาร์ดแวร์แบบแนวคิดเชิงวัตถุสำหรับการนำกลับมาใช้ [8] และงานวิจัยที่นำเสนอตัวอย่างของการนำแนวคิดการนำกลับมาใช้มาใช้ในการออกแบบดิจิทัล [15]

2.2.4 การสังเคราะห์วงจร การทวนสอบวงจรโดยการใช้แนวคิดเชิงวัตถุ

วิจัยซึ่งนำแนวคิดเชิงวัตถุมาประยุกต์กับการสังเคราะห์และการทวนสอบวงจร [7] ซึ่งการสังเคราะห์และทวนสอบวงจรเป็นขั้นตอนที่มีความสำคัญขั้นตอนหนึ่งในการออกแบบฮาร์ดแวร์ ในขั้นตอนนี้จะเป็นการตรวจสอบวงจรที่ได้ออกแบบว่ามีข้อผิดพลาด การทำงานและมีประสิทธิภาพการทำงานตรงตามความต้องการหรือไม่ ก่อนที่จะนำเอาแบบที่ได้ออกไว้ไปสร้างจริง

สำหรับวิทยานิพนธ์นี้ จะนำแนวคิดของงานในกลุ่มที่สามและสี่มาเป็นพื้นฐาน และเพิ่มเติมแนวคิดสำคัญจากแนวคิดเชิงวัตถุมาประกอบในการสร้างเครื่องมือสำหรับออกแบบวงจรตรรกะเชิงผสม เพื่อให้ได้ประโยชน์เพิ่มขึ้นจากเดิม

บทที่ 3
การออกแบบด้วยแนวคิดเชิงวัตถุ
ในการออกแบบวงจรตรรกะเชิงผสม

ความสำคัญของวิทยานิพนธ์นี้คือ การประยุกต์แนวคิดเชิงวัตถุทั้งสามลักษณะ คือ แนวคิดการห่อหุ้ม แนวคิดการถ่ายทอด และแนวคิดการนำกลับมาใช้ มาใช้ในการออกแบบฮาร์ดแวร์ ในส่วนของวงจรตรรกะเชิงผสมในระดับเกทและสร้างเครื่องมือเพื่อรองรับหลักการดังกล่าวเพื่อให้นักออกแบบฮาร์ดแวร์ได้มีทางเลือกใหม่เพิ่มเติม และช่วยลดเวลาและความซับซ้อน

แนวคิดเชิงวัตถุทั้งสามลักษณะ ได้แก่ แนวคิดการห่อหุ้ม แนวคิดการถ่ายทอด และ แนวคิดการนำกลับไปใช้ และฮาร์ดแวร์วงจรระดับเกท ได้แก่ แอนด์ ออร์ บัฟเฟอร์ ตัวผกผัน แนนด์ นอร์ ออร์เฉพาะ และออร์ไม่เฉพาะ จะได้รับการวิเคราะห์และประยุกต์ใช้ดังนี้

3.1 แนวคิดการห่อหุ้มกับเกท

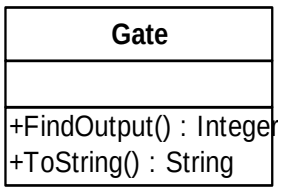
คือ การนำเอาข้อมูลมาผนวกรวมไว้กับการกระทำที่เกี่ยวข้องกับข้อมูลให้เป็นหนึ่งเดียว ที่เรียกว่า คลาสและวัตถุ เกทพื้นฐานจำนวนแปดชนิดที่ใช้ในการออกแบบวงจรตรรกะเชิงผสม เมื่อนำไปใช้ในการออกแบบวงจรตรรกะเชิงผสม เกทแต่ละเกทที่ได้นำไปออกแบบจะประกอบไปด้วยค่าความจริงของอินพุต และ ค่าความจริงของเอาต์พุต โดยที่ค่าความจริงของเอาต์พุตจะคำนวณได้จากพฤติกรรมของเกทแต่ละชนิด ดังนั้นแนวคิดการห่อหุ้มสามารถนำมาใช้ออกแบบเกทพื้นฐานทั้งแปดชนิดเพื่อความสะดวกในการนำเกทมาใช้ และเพิ่มความถูกต้องในขั้นตอนการออกแบบวงจรตรรกะเชิงผสม งานวิทยานิพนธ์นี้ได้ออกแบบคลาสชื่อ Gate ดังแสดงในรูปที่ 3.1 โดยคลาส Gate ประกอบด้วยสองพฤติกรรม คือ

1) FindOutput() คือ พฤติกรรมสำหรับคำนวณหาค่าความจริงของเอาต์พุต

2) ToString() คือ พฤติกรรมสำหรับแสดงผลลัพธ์ที่ได้จากการคำนวณหาค่าความจริงจากพฤติกรรม FindOutput() ในรูปแบบของสายอักขระ (String)

3.2 แนวคิดการถ่ายทอดกับเกท

คือ การที่ข้อมูลและพฤติกรรมทั้งหมดที่มีอยู่ในคลาสพ่อ ได้ถ่ายทอดไปยังคลาสลูก ซึ่งการออกแบบวงจรตรรกะเชิงผสมนั้น เกทจำนวนแปดชนิดที่ได้นำมาพิจารณาสามารถออกแบบเป็น



รูปที่ 3.1 คลาส Gate

คลาสของเกทแต่ละชนิดตามแนวคิดการห่อหุ้ม ซึ่งมีข้อมูลเป็นจำนวนอินพุต ค่าความจริงของอินพุตและเอาต์พุต และมีพฤติกรรมสำหรับคำนวณหาค่าความจริงของเอาต์พุต แต่เนื่องจากเกทแต่ละชนิดมีพฤติกรรมของการหาค่าความจริงของเอาต์พุตที่แตกต่างกัน ดังนั้นได้ออกแบบคลาสของเกททั้งแปดชนิดโดยรับการถ่ายทอดพฤติกรรมการหาค่าความจริงของเอาต์พุตจากเกท และเปลี่ยนแปลงเฉพาะรายละเอียดของการหาค่าความจริงจากพฤติกรรมที่ได้รับการถ่ายทอดมาของเกทแต่ละชนิด ซึ่งช่วยลดเวลาการออกแบบคลาสของเกททั้งแปดชนิด โดยไม่ต้องออกแบบคลาสทั้งแปดคลาสตั้งแต่ต้น ซึ่งเกทจำนวนแปดชนิดที่ได้นำมาพิจารณาในวิทยานิพนธ์นี้ มีพฤติกรรมที่สัมพันธ์กับคลาส Gate คือ หาค่าความจริงของเอาต์พุต และแสดงผลลัพธ์ที่ได้จากการคำนวณหาค่าความจริงในรูปแบบของสายอักขระ จึงออกแบบคลาส AndGate, OrGate, NotGate, NandGate, NorGate, XorGate, XNorGate และ BufferGate ได้รับการถ่ายทอดคุณสมบัติพฤติกรรม FindOutput() และ ToString() มาจากคลาส Gate และกำหนดให้คลาส AndGate, OrGate, NandGate, NorGate, XorGate และ XNorGate มีข้อมูล ดังต่อไปนี้

1) i และ j เป็นข้อมูลชนิดจำนวนเต็ม ทำหน้าที่เป็นดัชนีชี้ตำแหน่งของแถวลำดับ (Array)
2) numberOfInputs เป็นข้อมูลชนิดจำนวนเต็ม ทำหน้าที่เก็บจำนวนอินพุตของวัตถุนั้นๆ
3) inputValue(8) เป็นแถวลำดับชนิดจำนวนเต็มที่มีขนาดเก้าแถว ทำหน้าที่เก็บค่าความจริงของอินพุต ซึ่งสามารถรองรับได้มากที่สุดเก้าอินพุต เนื่องจากการออกแบบวงจรตรรกะเชิงผสมในระดับเกท โดยส่วนใหญ่ เกทหนึ่งๆ จะมีอินพุตมากที่สุดเป็นจำนวนเก้าอินพุต
4) output เป็นข้อมูลชนิดจำนวนเต็ม ทำหน้าที่เก็บค่าความจริงของเอาต์พุต ซึ่งเป็นผลลัพธ์ที่ได้จากการคำนวณค่าความจริงของเกทจากเมทธอด FindOutput()

พฤติกรรมของคลาส AndGate, OrGate, NandGate, NorGate, XorGate และ XNorGate ประกอบด้วย

1) Get nNumberOfInputs() ทำหน้าที่คืนค่าจำนวนอินพุตของเกทชนิดจำนวนเต็มของข้อมูล numberOfInputs เพื่อใช้ในการคำนวณหาค่าความจริงของเกท

2) Let nNumberOfInputs(ByVal numIn As Integer) ทำหน้าที่กำหนดจำนวนอินพุตของเกทให้กับข้อมูล numberOfInputs โดยผ่านพารามิเตอร์ชื่อ numIn ซึ่งเป็นข้อมูลชนิดจำนวนเต็ม เพื่อใช้ในการคำนวณหาค่าความจริงของเกท
3) Get IinputValue() ทำหน้าที่คืนค่าความจริงชนิดจำนวนเต็มที่เป็นอินพุตของเกทของข้อมูล inputValue(8) เพื่อใช้ในการคำนวณหาค่าความจริงของเกท
4) Let IinputValue(ByVal in1 As Integer) ทำหน้าที่กำหนดค่าความจริงอินพุตของเกทของข้อมูล inputValue(8) โดยผ่านพารามิเตอร์ชื่อ in1 ซึ่งเป็นข้อมูลชนิดจำนวนเต็ม เพื่อใช้ในการคำนวณหาค่าความจริงของเกท
5) Get Ooutput() ทำหน้าที่คืนค่าความจริงชนิดจำนวนเต็มที่เป็นเอาต์พุตของเกทของข้อมูล output เพื่อใช้ในการคำนวณหาค่าความจริงของเกท
6) Let Ooutput(ByVal out As Integer) ทำหน้าที่กำหนดค่าความจริงของเอาต์พุตของเกทให้กับข้อมูล output โดยผ่านพารามิเตอร์ชื่อ out ซึ่งเป็นข้อมูลชนิดจำนวนเต็ม เพื่อใช้ในการคำนวณหาค่าความจริงของเกท
7) FindOutput() ทำหน้าที่คำนวณหาค่าความจริงของเอาต์พุตของเกท
8) ToString() ทำหน้าที่แสดงผลลัพธ์ที่ได้จากการคำนวณหาค่าความจริงจากเมทธอด

สำหรับคลาส NotGate และ BufferGate ถูกกำหนดให้มีข้อมูลดังต่อไปนี้

1) input1 เป็นข้อมูลชนิดจำนวนเต็ม ทำหน้าที่เก็บค่าความจริงของอินพุตของเกทชนิดตัวผกผันและบัฟเฟอร์ ซึ่งเกททั้งสองชนิดจะมีจำนวนอินพุตจำกัดได้เพียงหนึ่งอินพุตเท่านั้น
2) output เป็นข้อมูลชนิดจำนวนเต็ม ทำหน้าที่เก็บค่าความจริงของเอาต์พุต ซึ่งเป็นผลลัพธ์ที่ได้จากการคำนวณค่าความจริงของเกทจากเมทธอด FindOutput()

พฤติกรรมของคลาส NotGate และ BufferGate ประกอบด้วย

1) Get Iinput1() ทำหน้าที่คืนค่าความจริงของอินพุตของเกทชนิดจำนวนเต็มของข้อมูล input1 เพื่อใช้ในการคำนวณหาค่าความจริงของเกท
2) Let Iinput1(ByVal in1 As Integer) ทำหน้าที่กำหนดค่าความจริงของอินพุตของเกทให้กับข้อมูล Iinput1 โดยผ่านพารามิเตอร์ชื่อ in1 ซึ่งเป็นข้อมูลชนิดจำนวนเต็ม เพื่อใช้ในการคำนวณหาค่าความจริงของเกท
3) Get Ooutput() ทำหน้าที่คืนค่าความจริงชนิดจำนวนเต็มที่เป็นเอาต์พุตของเกทของข้อมูล output เพื่อใช้ในการคำนวณหาค่าความจริงของเกท

4) Let Ooutput(ByVal out As Integer) ทำหน้าที่กำหนดค่าความจริงของเอาต์พุตของเกทให้กับข้อมูล output ซึ่งเป็นข้อมูลชนิดจำนวนเต็ม โดยผ่านพารามิเตอร์ชื่อ out เพื่อใช้ในการคำนวณหาค่าความจริงของเกท

5) FindOutput() ทำหน้าที่คำนวณหาค่าความจริงของเอาต์พุตของเกท

6) ToString() ทำหน้าที่แสดงผลลัพธ์ที่ได้จากการคำนวณหาค่าความจริงจากเมทธอด

โครงสร้างของคลาส AndGate, OrGate, NotGate, NandGate, NorGate, XorGate, XNorGate และ BufferGate ที่ได้รับการถ่ายทอดพฤติกรรมมาจากคลาส Gate ตามที่ได้ออกแบบขึ้นในวิทยานิพนธ์นี้ มีลักษณะ ดังแสดงในรูปที่ 3.2

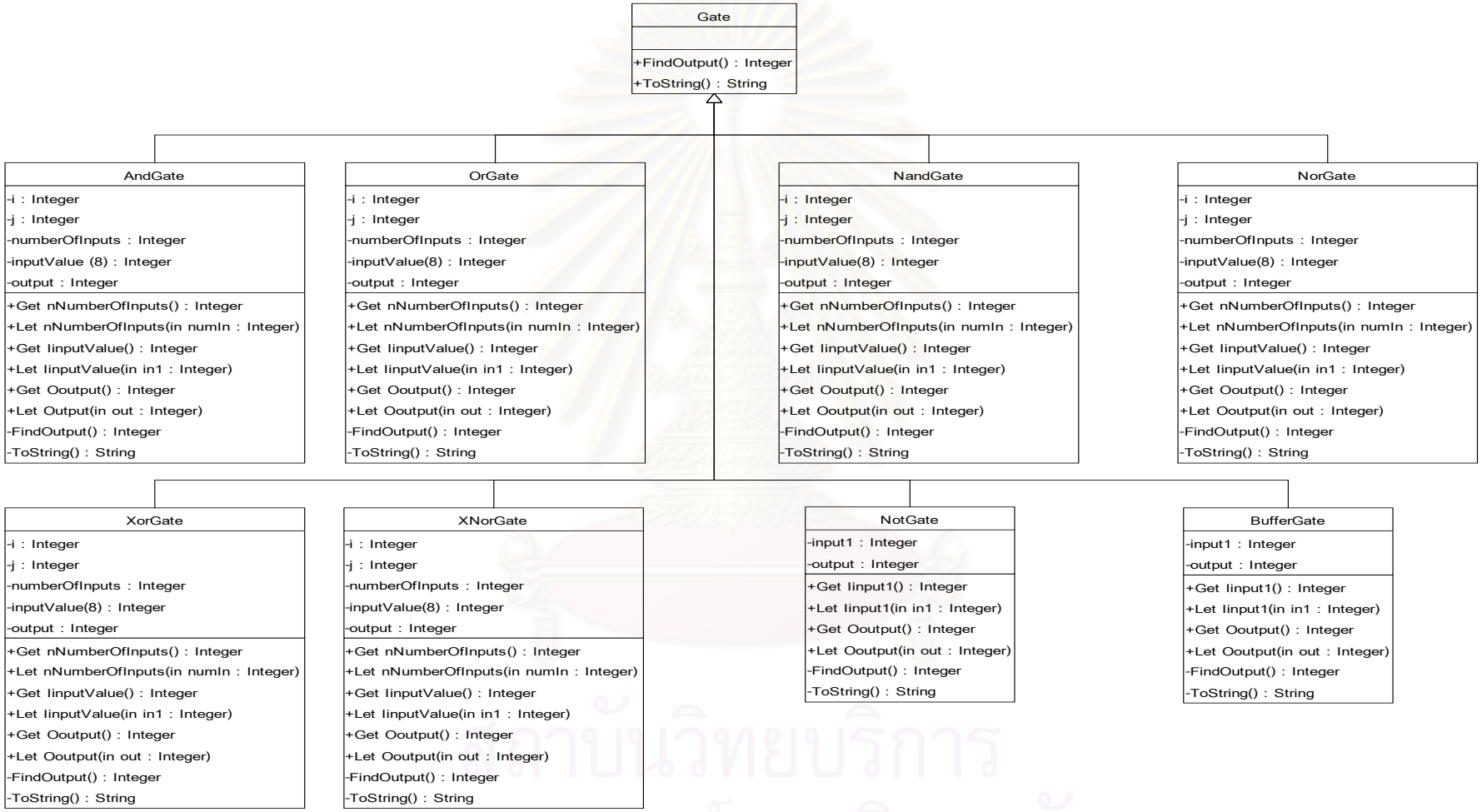
เกทแต่ละชนิดมีพฤติกรรมที่แตกต่างกันในการหาค่าความจริงของเอาต์พุต [20] คือ

- แอนด์ จะให้ค่าความจริงเป็นจริงเมื่อ อินพุตทุกตัวมีค่าความจริง เป็นจริง
- ออร์ จะให้ค่าความจริงเป็นเท็จเมื่อ อินพุตทุกตัวมีค่าความจริงเป็นเท็จ
- ตัวผกผัน จะให้ค่าความจริงเป็นค่าที่ตรงข้ามกับค่าความจริงของอินพุต
- แนนด์ จะให้ค่าความจริงเป็นเท็จเมื่อ อินพุตทุกตัวมีค่าความจริงเป็นจริง
- นอร์ จะให้ค่าความจริงเป็นจริงเมื่อ อินพุตทุกตัวมีค่าความจริงเป็นเท็จ
- ออร์เฉพาะ จะให้ค่าความจริงเป็นจริงเมื่อ จำนวนค่าความจริงของอินพุตที่มีค่าเป็นจริงเป็นจำนวนคี่
- ออร์ไม่เฉพาะ จะให้ค่าความจริงเป็นจริงเมื่อ จำนวนค่าความจริงของอินพุตที่มีค่าเป็นจริงเป็นจำนวนคู่
- บัฟเฟอร์ จะให้ค่าความจริงตามค่าความจริงของอินพุต

ดังนั้นคลาสของเกททั้งแปดชนิด ที่ได้รับการถ่ายทอดพฤติกรรมแต่มีการเปลี่ยนแปลงรายละเอียดพฤติกรรมการหาค่าความจริงของเอาต์พุตซึ่งแตกต่างกันตามชนิดของเกท แสดงให้เห็นถึงคุณสมบัติการโอเวอร์ไรด์ภายใต้คุณสมบัติการถ่ายทอด

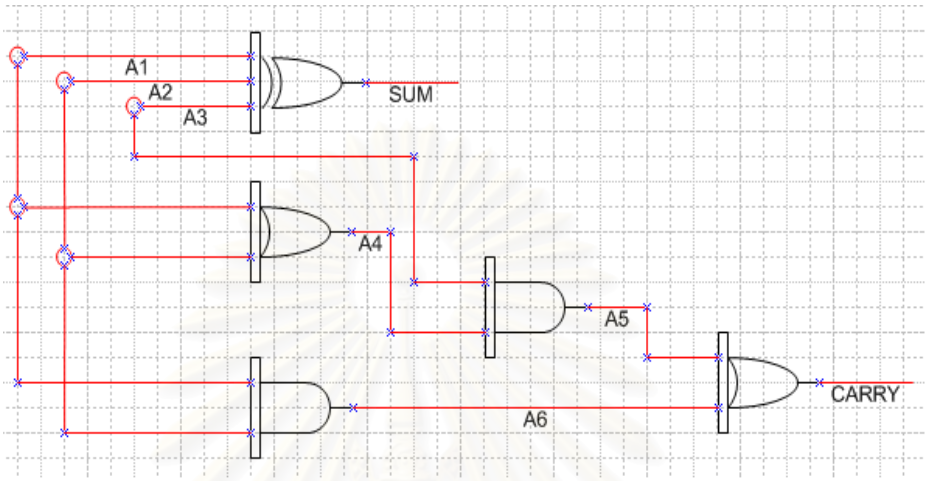
3.3 แนวคิดการนำกลับมาใช้กับเกท

วิทยานิพนธ์นี้ได้นำแนวคิดเชิงวัตถุมาประยุกต์ใช้กับการออกแบบวงจรตรรกะเชิงผสม โดยวงจรที่ถูกออกแบบขึ้นเพื่อนำกลับมาใช้จะถูกจัดเก็บในรูปแบบของไลบรารี คือ วงจรที่ผู้ออกแบบฮาร์ดแวร์ได้ออกแบบโดยมีวัตถุประสงค์ที่จะนำกลับไปใช้ วงจรดังกล่าวจะถูกนำไปสังเคราะห์วงจรแบบลักษณะของการนำกลับมาใช้ ถ้าการสังเคราะห์วงจรไม่พบข้อผิดพลาด วงจรดังกล่าวจะถูกนำไปสร้างเป็นวงจรบล็อก (Circuit Block) ซึ่งมีจำนวนบิตของอินพุตและเอาต์พุตหลักคงที่ และถูกจัดเก็บในรูปแบบของไลบรารี เพื่อที่ผู้ออกแบบฮาร์ดแวร์สามารถนำวงจร

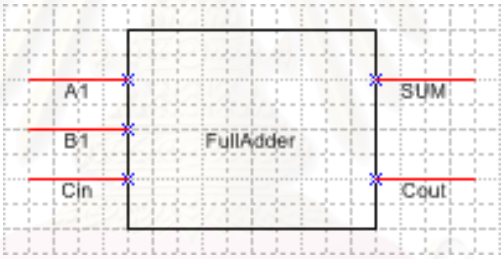


รูปที่ 3.2 คลาสของเกททั้งแปดชนิดกับการถ่ายทอดพฤติกรรมจากคลาส Gate

บล็อกกลับมาใช้ได้อีกตามความเหมาะสมของการออกแบบวงจรใดๆ โดยไม่ต้องเสียเวลาและแรงงานในการออกแบบวงจรบล็อกดังกล่าวอีกครั้งหนึ่ง ดังแสดงในรูปที่ 3.3 และ 3.4



รูปที่ 3.3 วงจรเต็มบวกที่ถูกสร้างขึ้นเป็นวงจรบล็อก



รูปที่ 3.4 การนำวงจรบล็อกของวงจรเต็มบวกกลับมาใช้

บทที่ 4
การออกแบบเครื่องมือสำหรับออกแบบวงจรตรรกะเชิงผสม แบบแนวคิดเชิงวัตถุ

การออกแบบวงจรตรรกะเชิงผสมในวิทยานิพนธ์นี้ได้นำแนวคิดเชิงวัตถุสามแนวคิด คือ 1) แนวคิดการห่อหุ้ม 2) แนวคิดการถ่ายทอด และ 3) แนวคิดการนำกลับมาใช้ มาใช้ตามที่ได้อธิบายไว้ในบทที่สาม ดังนั้นเพื่อที่จะแสดงการออกแบบวงจรตรรกะเชิงผสมด้วยแนวคิดเชิงวัตถุ จึงได้ออกแบบและพัฒนาเครื่องมือสำหรับออกแบบวงจรตรรกะเชิงผสมตามแบบแนวคิดเชิงวัตถุที่ได้ออกแบบไว้ในบทที่สาม สำหรับเนื้อหาของการออกแบบเครื่องมือสำหรับออกแบบวงจรตรรกะเชิงผสมแบบแนวคิดเชิงวัตถุสามารถแบ่งการอธิบายเนื้อหาออกเป็นสามส่วน ซึ่งได้แบ่งตามส่วนประกอบของเครื่องมือที่ได้สร้างขึ้นในวิทยานิพนธ์นี้ คือ

1) ส่วนออกแบบวงจรตรรกะเชิงผสมแบบกราฟิก

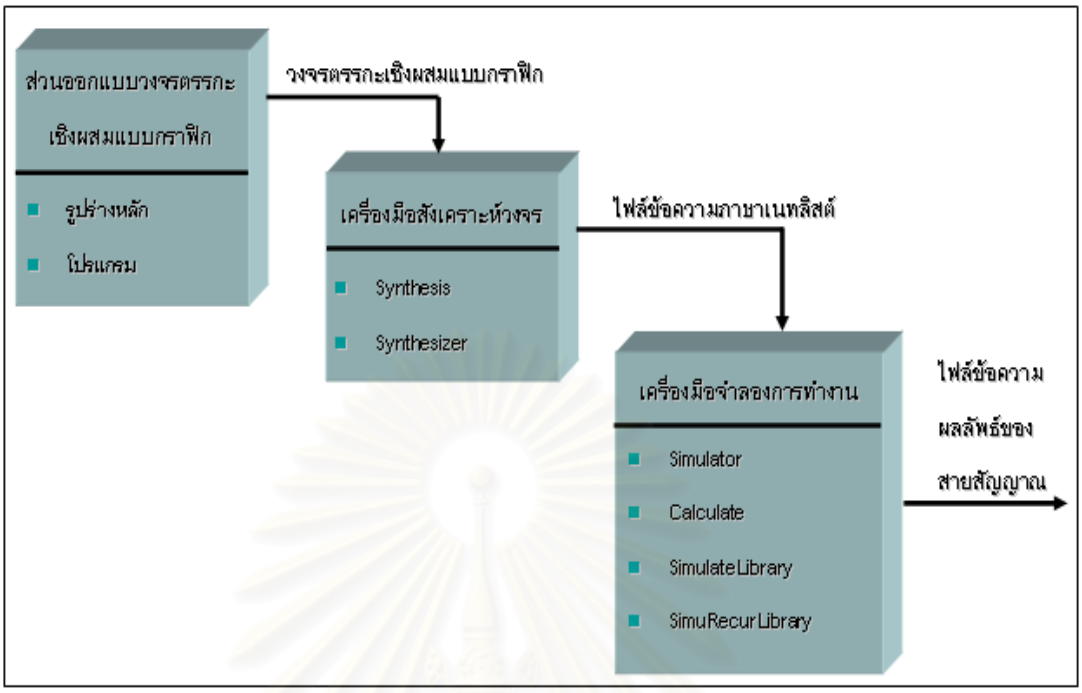
2) เครื่องมือสังเคราะห์วงจร

3) เครื่องมือจำลองการทำงาน

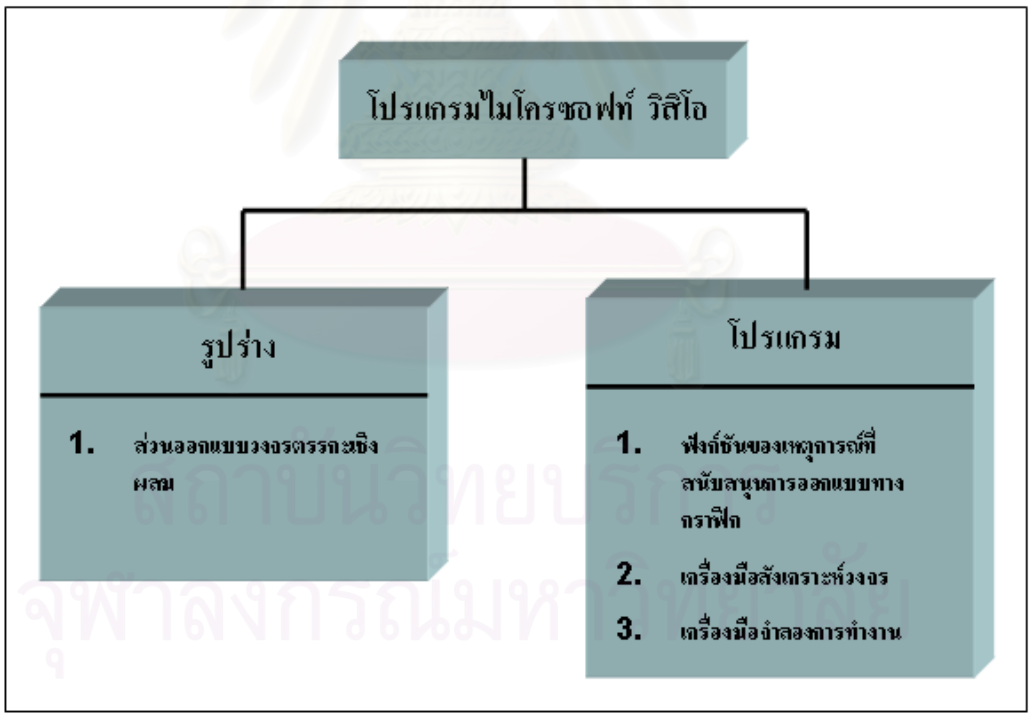
ดังแสดงในรูปที่ 4.1 โดยส่วนประกอบของเครื่องมือที่สร้างขึ้นในวิทยานิพนธ์นี้ได้พัฒนาขึ้นด้วยโปรแกรมภาษาวิชวลเบสิกสำหรับแอพลิเคชัน (Visual Basic for Application, VBA) ร่วมกับโปรแกรมสภาพแวดล้อมการพัฒนาไมโครซอฟท์ วิสิโอ เวอร์ชัน เอนเทอร์ไพร์ส อาร์คิเท็ค (Microsoft Visio, Enterprise Architect Version) ซึ่งโปรแกรมไมโครซอฟท์ วิสิโอ มีส่วนประกอบสองส่วนที่ใช้สำหรับการออกแบบ คือ ส่วนของรูปร่างและโปรแกรม ซึ่งส่วนประกอบของเครื่องมือที่สร้างขึ้นในวิทยานิพนธ์นี้เป็นส่วนประกอบของรูปร่างและโปรแกรม ดังแสดงในรูปที่ 4.2 สำหรับส่วนประกอบของเครื่องมือ ซึ่งประกอบด้วย ส่วนออกแบบวงจรตรรกะเชิงผสมแบบกราฟิก เครื่องมือสังเคราะห์วงจร และเครื่องมือจำลองการทำงาน

4.1 ส่วนออกแบบวงจรตรรกะเชิงผสมแบบกราฟิก

ส่วนออกแบบวงจรตรรกะเชิงผสมแบบกราฟิก ทำหน้าที่เป็นส่วนต่อประสานกับผู้ออกแบบฮาร์ดแวร์ (User Interface) เพื่อใช้ในการออกแบบวงจรตรรกะเชิงผสม รับคำสั่งและค่าต่างๆ ที่จำเป็นสำหรับการออกแบบวงจรตรรกะเชิงผสม ส่วนออกแบบวงจรตรรกะเชิงผสมแบบกราฟิกถูกพัฒนาขึ้นด้วยโปรแกรมภาษาวิชวลเบสิกสำหรับแอพลิเคชัน ร่วมกับโปรแกรมสภาพแวดล้อมการพัฒนาไมโครซอฟท์ วิสิโอ เวอร์ชัน เอนเทอร์ไพร์ส อาร์คิเท็ค เนื่องจาก



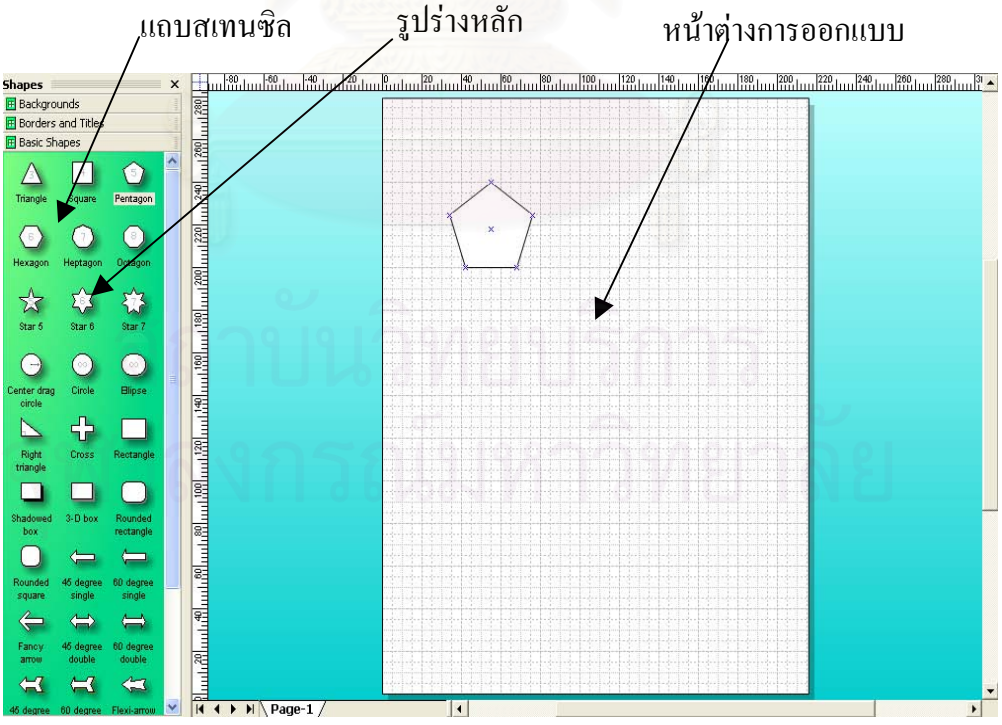
รูปที่ 4.1 ภาพรวมของเครื่องมือที่สร้างขึ้นในวิทยานิพนธ์



รูปที่ 4.2 ส่วนประกอบของเครื่องมือที่ได้พัฒนาขึ้นภายใต้โปรแกรมไมโครซอฟท์ วิสิโอ

โปรแกรมไมโครซอฟท์ วิสิโอ มีคุณสมบัติทางด้านรูปร่าง (Shape) และโปรแกรม ซึ่งช่วยสนับสนุนการแก้ไขปัญหาการออกแบบใดๆ ทางด้านกราฟิกได้เป็นอย่างดี คือ

1) *รูปร่าง* ไมโครซอฟท์ วิสิโอมีรูปร่างหลัก (Master shape) ชนิดพื้นฐานต่างๆ ที่โปรแกรมจัดสร้างขึ้นไว้แล้วและถูกจัดหมวดหมู่อยู่ใน สเทนซิล (Stencil) ซึ่งผู้ออกแบบฮาร์ดแวร์สามารถลากรูปร่างหลักที่ปรากฏอยู่ในแถบสเทนซิลมาปล่อยยังหน้าต่างการออกแบบ เพื่อออกแบบสิ่งใดๆ ตามที่ผู้ออกแบบฮาร์ดแวร์ต้องการ ดังแสดงในรูปที่ 4.3 ทำให้ผู้ออกแบบฮาร์ดแวร์มีความสะดวกในการออกแบบมากขึ้นเพราะว่าผู้ออกแบบฮาร์ดแวร์ไม่จำเป็นต้องวาดรูปร่างพื้นฐานที่ซ้ำซ้อนกันหลายๆ ครั้ง นอกจากนี้ผู้ออกแบบฮาร์ดแวร์ยังสามารถสร้างรูปร่างหลักและสร้างสเทนซิลเพื่อจัดเก็บรูปร่างหลักที่ผู้ออกแบบฮาร์ดแวร์สร้างขึ้นเอง เพื่อใช้ในงานออกแบบใดๆ ที่มีความเฉพาะเจาะจง เนื่องจากในวิทยานิพนธ์นี้ต้องการสร้างเครื่องมือสำหรับออกแบบวงจรตรรกะเชิงผสมซึ่งประยุกต์ใช้แนวคิดเชิงวัตถุ โดยส่วนออกแบบวงจรตรรกะเชิงผสมแบบกราฟิกนั้น ประกอบด้วยเกทพื้นฐานจำนวนแปดชนิดที่ได้มีการออกแบบใหม่เพื่อให้สอดคล้องกับการประยุกต์ใช้แนวคิดเชิงวัตถุ ซึ่งเกทพื้นฐานจำนวนแปดชนิดนี้จะถูกสร้างขึ้นเป็นรูปร่างหลักและถูกจัดเก็บไว้เป็นหมวดหมู่ในสเทนซิลที่ถูกสร้างขึ้น ดังนั้นโปรแกรมไมโครซอฟท์ วิสิโอสามารถสนับสนุนส่วนออกแบบวงจรตรรกะเชิงผสมแบบกราฟิกได้เป็นอย่างดี



รูปที่ 4.3 แถบสเทนซิล รูปร่างหลัก และหน้าต่างการออกแบบ

2) *โปรแกรม* ภาษาวิชวลเบสิกสำหรับแอพลิเคชันสามารถช่วยสร้างการออกแบบแก่ผู้ออกแบบฮาร์ดแวร์โดยตรง นอกจากการที่ผู้ออกแบบฮาร์ดแวร์จะลากรูปร่างหลักจากสเทนซิลมาปล่อยบนหน้าต่างการออกแบบแล้ว ผู้ออกแบบฮาร์ดแวร์ยังสามารถเขียนโปรแกรมสั่งให้รูปร่างหลักถูกวาดลงบนหน้าต่างการออกแบบได้ นอกจากโปรแกรมจะช่วยผู้ออกแบบฮาร์ดแวร์ในเรื่องของการออกแบบแล้ว โปรแกรมยังสามารถถ่ายโอนข้อมูลของสิ่งที่ถูกออกแบบบนหน้าต่างการออกแบบกับข้อมูลจากโปรแกรมภายนอก หรือข้อมูลของสิ่งที่ถูกออกแบบบนหน้าต่างการออกแบบด้วยกัน และโปรแกรมยังสามารถถูกพัฒนาเพื่อนำข้อมูลของสิ่งที่ถูกออกแบบบนหน้าต่างการออกแบบไปประมวลผลตามที่ผู้ออกแบบฮาร์ดแวร์ต้องการ ดังนั้นโปรแกรมไมโครซอฟท์ วิสิโอสามารถสนับสนุนส่วนออกแบบวงจรตรรกะเชิงผสมแบบกราฟิกได้เป็นอย่างดี เนื่องจากโปรแกรมสามารถถ่ายโอนข้อมูลของรูปร่างที่ถูกออกแบบบนหน้าต่างการออกแบบไปยังข้อมูลของอีกรูปร่างหนึ่งที่อยู่บนหน้าต่างการออกแบบได้ และโปรแกรมยังสามารถพัฒนาตัวสังเคราะห์วงจร และตัวจำลองการทำงานเพื่อนำรูปร่าง และข้อมูลของรูปร่างทั้งหมดที่ถูกออกแบบไว้บนหน้าต่างการออกแบบไปประมวลผลได้เป็นอย่างดีเนื่องจากโปรแกรมสามารถถ่ายโอนข้อมูลของรูปร่างที่ถูกออกแบบบนหน้าต่างการออกแบบไปยังข้อมูลของอีกรูปร่างหนึ่งที่อยู่บนหน้าต่างการออกแบบได้ และโปรแกรมยังสามารถพัฒนาตัวสังเคราะห์วงจรและตัวจำลองการทำงานเพื่อนำรูปร่างและข้อมูลของรูปร่างทั้งหมดที่ถูกออกแบบไว้บนหน้าต่างการออกแบบไปประมวลผลได้

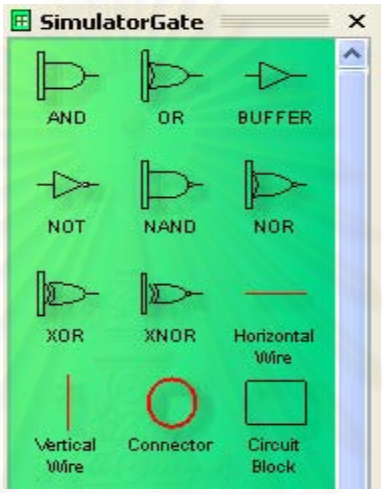
ส่วนออกแบบวงจรตรรกะเชิงผสมแบบกราฟิกได้รับการออกแบบขึ้นในวิทยานิพนธ์นี้ประกอบด้วยสองส่วน ดังนี้

4.1.1 การออกแบบสเทนซิลและรูปร่างหลัก

4.1.1.1 การออกแบบสเทนซิล

สเทนซิล มีหน้าที่สำหรับเก็บรูปร่างหลักชนิดต่างๆ ที่ใช้ในการออกแบบ สเทนซิลสร้างขึ้นโดยเลือก File -> Stencils -> New Stencil จากหน้าต่างของการออกแบบ จากนั้นให้กำหนดชื่อให้กับสเทนซิลและลากรูปร่างที่ออกแบบมาปล่อยยังสเทนซิล จะปรากฏรูปร่างหลักของรูปร่างที่ได้ออกแบบไว้บนสเทนซิล ดังนั้นส่วนออกแบบวงจรตรรกะเชิงผสมแบบกราฟิกนี้ จึงได้สร้างสเทนซิลที่มีชื่อว่า SimulatorGate สำหรับเก็บรูปร่างหลักที่ถูกสร้างขึ้นในวิทยานิพนธ์นี้

ประกอบไปด้วยเกทพื้นฐานจำนวนแปดเกท สายสัญญาณแนวตั้ง (Vertical Wire) สายสัญญาณแนวนอน (Horizontal Wire) จุดเชื่อมต่อ (Connector) และวงจรบล็อก (Circuit Block) ซึ่งผู้ออกแบบฮาร์ดแวร์สามารถลากรูปร่างหลักดังกล่าวไปปล่อยในหน้าต่างการออกแบบเพื่อออกแบบวงจรตรรกะเชิงผสมที่ต้องการ โดยที่สเทนซิล SimulatorGate มีลักษณะดังรูปที่ 4.4

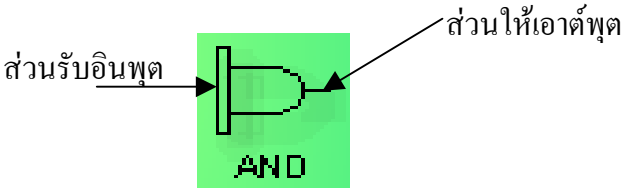


รูปที่ 4.4 สเทนซิล SimulatorGate สำหรับการออกแบบวงจรตรรกะเชิงผสม

4.1.1.2 การออกแบบรูปร่างหลัก

รูปร่างหลัก คือ สัญลักษณ์ที่ถูกเก็บไว้ในสเทนซิลเพื่อใช้ในการออกแบบในหน้าต่างการออกแบบโดยหลักการลากแล้วปล่อย สำหรับรูปร่างหลักที่ถูกออกแบบและสร้างขึ้นมาในส่วนออกแบบวงจรตรรกะเชิงผสมแบบกราฟิกนี้ประกอบไปด้วยสิบสองรูปร่างหลัก คือ

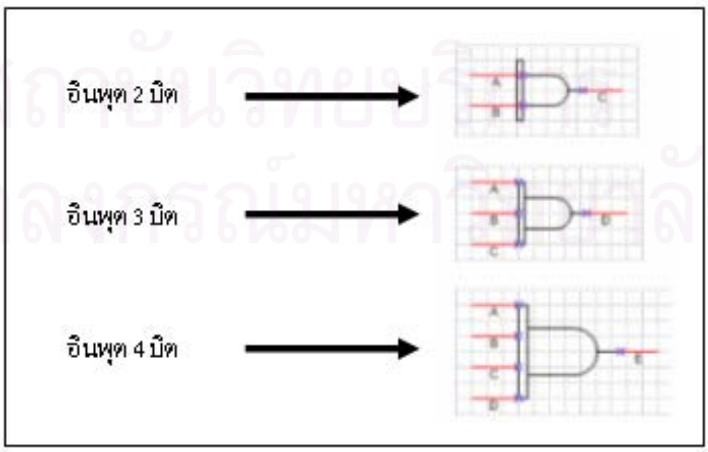
1) แอนด์ (AND) ถูกออกแบบและสร้างขึ้นเพื่อใช้เป็นสัญลักษณ์แทนแอนด์เกทเพื่อคำนวณค่าความจริงของแอนด์ ซึ่ง แอนด์สร้างขึ้นโดยสร้างรูปร่างของแอนด์เกทซึ่งมีลักษณะดังรูปที่ 4.5 ในหน้าต่างของการออกแบบแอนด์เกทที่ออกแบบขึ้นประกอบด้วย เส้นตรง (Line Tool) และเส้นโค้ง (Arc Tool) ซึ่งเป็นเครื่องมือพื้นฐานในการออกแบบของโปรแกรมไมโครซอฟท์ วิสิโอ จากนั้นจึงลากแอนด์เกทที่ได้ออกแบบขึ้นไปปล่อยยังสเทนซิลชื่อ SimulatorGate และกำหนดชื่อของรูปร่างหลักดังกล่าวให้มีชื่อว่าแอนด์



รูปที่ 4.5 รูปร่างหลักของแอนด์

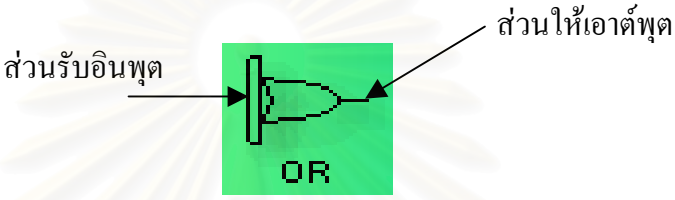
รูปร่างหลักแอนด์ ประกอบด้วยสองส่วนประกอบด้วยกัน คือ

- ส่วนรับอินพุต จะถูกออกแบบให้มีรูปร่างเป็นสี่เหลี่ยมอยู่ติดกับด้านหลังของลักษณะรูปร่างเกทเพื่อใช้ในการรับสายสัญญาณอินพุตที่จะถูกเชื่อมต่อเข้ากับแอนด์โดยผู้ออกแบบสามารถเชื่อมต่อสายสัญญาณอินพุตเข้ากับส่วนรับอินพุตได้ในจำนวนไม่จำกัดแต่ต้องมีจำนวนอินพุตตั้งแต่สอง อินพุตขึ้นไป ดังแสดงตัวอย่างการเชื่อมต่อสายสัญญาณอินพุตสอง สาม และสี่อินพุตในรูปที่ 4.6 โดยที่ขนาดของรูปร่างสามารถขยายได้เมื่อรูปร่างหลักถูกนำไปใช้ในการออกแบบในหน้าต่างการออกแบบ แต่จำนวนอินพุตที่ตัวจำลองการทำงานสามารถคำนวณค่าความจริงของแอนด์นั้น อยู่ที่เก้าอินพุต เนื่องจากจำนวนอินพุตของเกทชนิดนี้ที่ใช้ในการออกแบบวงจร ISCAS85 มีจำนวนไม่เกินเก้าอินพุต [24]
- ส่วนให้เอาต์พุต สายสัญญาณที่ถูกเชื่อมต่อเข้ากับส่วนให้เอาต์พุตจะมีจำนวนเพียงหนึ่งสายสัญญาณ เนื่องจากเอาต์พุตของแอนด์เกทจะมีหนึ่งเอาต์พุตเท่านั้น และสายสัญญาณดังกล่าวจะถูกกำหนดให้เป็นสายสัญญาณเอาต์พุตของแอนด์



รูปที่ 4.6 รูปร่างหลักของแอนด์ที่ส่วนรับอินพุตเป็นจำนวนสอง สาม และสี่อินพุต

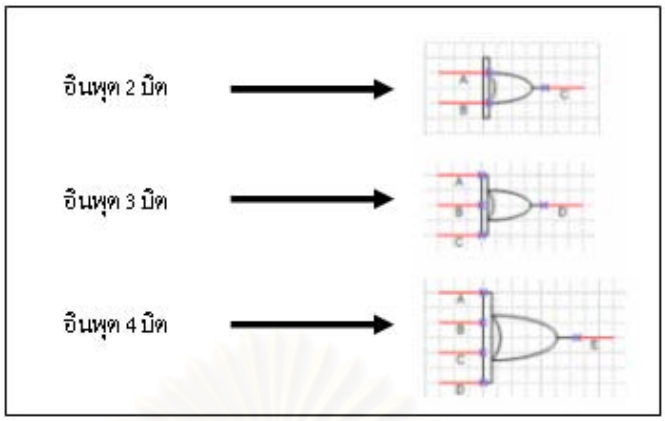
2) ออร์ (OR) ถูกออกแบบและสร้างขึ้นเพื่อใช้เป็นสัญลักษณ์แทนออร์เกทเพื่อคำนวณค่าความจริงของออร์ ซึ่งออร์เกทสร้างขึ้นโดยสร้างรูปร่างของออร์เกทซึ่งมีลักษณะดังรูปที่ 4.7 ในหน้าต่างของการออกแบบออร์เกทที่ออกแบบขึ้นประกอบด้วย เส้นตรงและเส้นโค้ง ซึ่งเป็นเครื่องมือพื้นฐานในการออกแบบของโปรแกรมไมโครซอฟท์ วิสิโอ จากนั้นจึงลากออร์เกทที่ได้ออกแบบขึ้นไปปล่อยยังสเทนซิล SimulatorGate และกำหนดชื่อของรูปร่างหลักดังกล่าวให้มีชื่อว่าออร์



รูปที่ 4.7 รูปร่างหลักของออร์

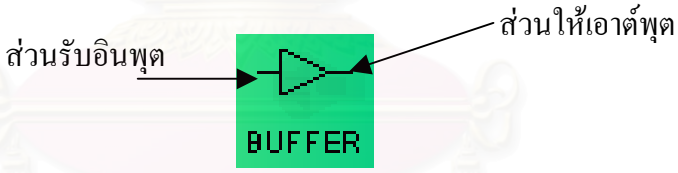
รูปร่างหลักออร์ ประกอบด้วยสองส่วนประกอบด้วยกัน คือ

- ส่วนรับอินพุต จะถูกออกแบบให้มีรูปร่างเป็นสี่เหลี่ยมอยู่ติดกับด้านหลังของลักษณะรูปร่างเกทเพื่อใช้ในการรับสายสัญญาณอินพุตที่จะถูกเชื่อมต่อเข้ากับออร์โดยผู้ออกแบบสามารถเชื่อมต่อสายสัญญาณอินพุตเข้ากับส่วนรับอินพุตได้ในจำนวนไม่จำกัดแต่ต้องมีจำนวนอินพุตตั้งแต่สองอินพุตขึ้นไป ดังแสดงตัวอย่างการเชื่อมต่อสายสัญญาณอินพุตสอง สามและสี่อินพุตในรูปที่ 4.8 โดยที่ขนาดของรูปร่างสามารถขยายได้เมื่อรูปร่างหลักถูกนำไปใช้ในการออกแบบในหน้าต่างการออกแบบ แต่จำนวนอินพุตที่ตัวจำลองการทำงานสามารถคำนวณค่าความจริงของออร์นั้น อยู่ที่เก้าอินพุต เนื่องจากจำนวนอินพุตของเกทชนิดนี้ที่ใช้ในการออกแบบวงจร ISCAS85 มีจำนวนไม่เกินเก้าอินพุต [24]
- ส่วนให้เอาต์พุต สายสัญญาณที่ถูกเชื่อมต่อเข้ากับส่วนให้เอาต์พุตจะมีจำนวนเพียงหนึ่งสายสัญญาณ เนื่องจากเอาต์พุตของออร์เกทจะมีหนึ่งเอาต์พุตเท่านั้น และสายสัญญาณดังกล่าวจะถูกกำหนดให้เป็นสายสัญญาณเอาต์พุตของออร์



รูปที่ 4.8 รูปร่างหลักของออร์ ที่ส่วนรับอินพุตเป็นจำนวนสอง สาม และสี่อินพุต

3) บัฟเฟอร์ (BUFFER) ถูกออกแบบและสร้างขึ้นเพื่อใช้เป็นสัญลักษณ์แทนบัฟเฟอร์เกทเพื่อคำนวณค่าความจริงของบัฟเฟอร์ ซึ่งบัฟเฟอร์เกทสร้างขึ้นโดยสร้างรูปร่างของบัฟเฟอร์เกทซึ่งมีลักษณะดังรูปที่ 4.9 ในหน้าต่างของการออกแบบบัฟเฟอร์เกทที่ออกแบบขึ้นประกอบด้วย เส้นตรง ซึ่งเป็นเครื่องมือพื้นฐานในการออกแบบของโปรแกรมไมโครซอฟท์ วิสิโอ จากนั้นจึงลากบัฟเฟอร์เกทที่ได้ออกแบบขึ้นไปปล่อยยังสเทนซิล SimulatorGate และกำหนดชื่อของรูปร่างหลักดังกล่าวให้มีชื่อว่าบัฟเฟอร์

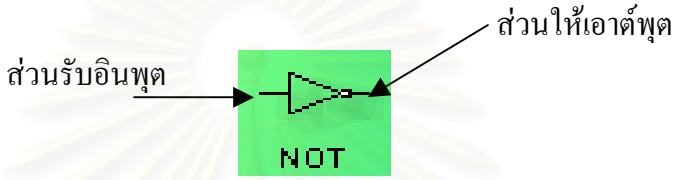


รูปที่ 4.9 รูปร่างหลักของบัฟเฟอร์

รูปร่างหลักบัฟเฟอร์ ประกอบด้วยสองส่วนประกอบด้วยกัน คือ

- ส่วนรับอินพุต สายสัญญาณที่ถูกเชื่อมต่อเข้ากับส่วนรับอินพุตจะมีจำนวนเพียงหนึ่งสายสัญญาณ เนื่องจากอินพุตของบัฟเฟอร์เกทจะมีเพียงหนึ่งอินพุตเท่านั้น และสายสัญญาณดังกล่าวจะถูกกำหนดให้เป็นสายสัญญาณอินพุตของบัฟเฟอร์
- ส่วนให้เอาต์พุต สายสัญญาณที่ถูกเชื่อมต่อเข้ากับส่วนให้เอาต์พุตจะมีจำนวนเพียงหนึ่งสายสัญญาณ เนื่องจากเอาต์พุตของบัฟเฟอร์เกทจะมีหนึ่งเอาต์พุตเท่านั้น และสายสัญญาณดังกล่าวจะถูกกำหนดให้เป็นสายสัญญาณเอาต์พุตของบัฟเฟอร์

4) ตัวผกผัน (NOT) ถูกออกแบบและสร้างขึ้นเพื่อใช้เป็นสัญลักษณ์แทนตัวผกผันเพื่อคำนวณค่าความจริงของตัวผกผัน ซึ่งตัวผกผันสร้างขึ้นโดยสร้างรูปร่างของตัวผกผันซึ่งมีลักษณะดังรูปที่ 4.10 ในหน้าต่างของการออกแบบ ตัวผกผันที่ออกแบบขึ้นประกอบด้วย เส้นตรงและเส้นโค้ง ซึ่งเป็นเครื่องมือพื้นฐานในการออกแบบของโปรแกรมไมโครซอฟท์ วิสิโอ จากนั้นจึงลาก ตัวผกผันที่ได้ออกแบบขึ้นไปปล่อยยังสเทนซิล SimulatorGate และกำหนดชื่อของรูปร่างหลักดังกล่าวให้มีชื่อว่าตัวผกผัน

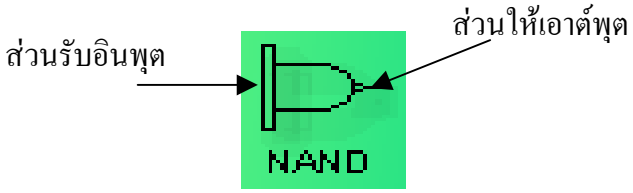


รูปที่ 4.10 รูปร่างหลักของตัวผกผัน

รูปร่างหลักตัวผกผัน ประกอบด้วยสองส่วนประกอบด้วยกัน คือ

- ส่วนรับอินพุต สายสัญญาณที่ถูกเชื่อมต่อเข้ากับส่วนรับอินพุตจะมีจำนวนเพียงหนึ่งสายสัญญาณ เนื่องจากอินพุตของตัวผกผันจะมีเพียงหนึ่งอินพุตเท่านั้น และสายสัญญาณดังกล่าวจะถูกกำหนดให้เป็นสายสัญญาณอินพุตของตัวผกผัน
- ส่วนให้เอาต์พุต สายสัญญาณที่ถูกเชื่อมต่อเข้ากับส่วนให้เอาต์พุตจะมีจำนวนเพียงหนึ่งสายสัญญาณ เนื่องจากเอาต์พุตของตัวผกผันจะมีหนึ่งเอาต์พุตเท่านั้น และสายสัญญาณดังกล่าวจะถูกกำหนดให้เป็นสายสัญญาณเอาต์พุตของตัวผกผัน

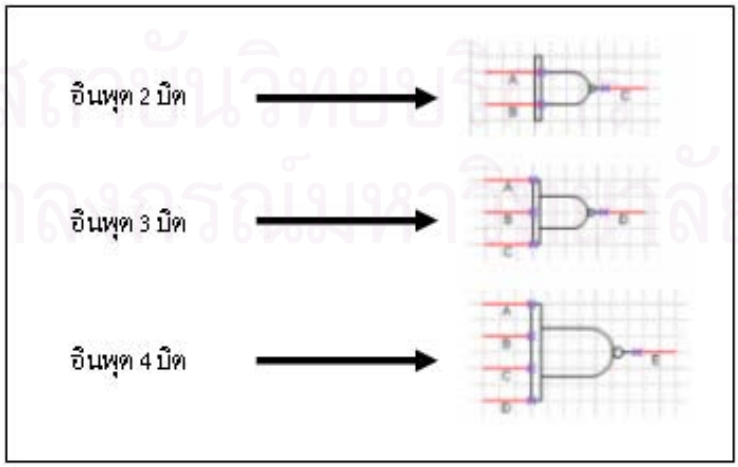
5) แนนด์ (NAND) ถูกออกแบบและสร้างขึ้นเพื่อใช้เป็นสัญลักษณ์แทนแนนด์เกทเพื่อคำนวณค่าความจริงของแนนด์ ซึ่งแนนด์เกทสร้างขึ้นโดยสร้างรูปร่างของแนนด์เกทซึ่งมีลักษณะดังรูปที่ 4.11 ในหน้าต่างของการออกแบบ แนนด์เกทที่ออกแบบขึ้นประกอบด้วย เส้นตรงและเส้นโค้ง ซึ่งเป็นเครื่องมือพื้นฐานในการออกแบบของโปรแกรมไมโครซอฟท์ วิสิโอ จากนั้นจึงลากแนนด์เกทที่ได้ออกแบบขึ้นไปปล่อยยังสเทนซิล SimulatorGate และกำหนดชื่อของรูปร่างหลักดังกล่าวให้มีชื่อว่าแนนด์



รูปที่ 4.11 รูปร่างหลักของแนนด์

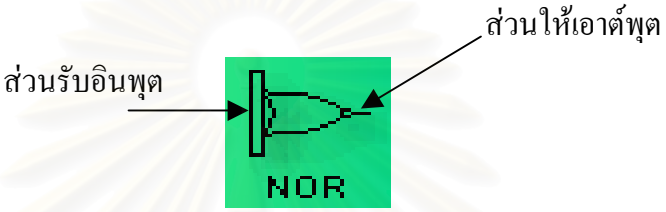
รูปร่างหลักแนนด์ ประกอบด้วยสองส่วนประกอบด้วยกัน คือ

- ส่วนรับอินพุต จะถูกออกแบบให้มีรูปร่างเป็นสี่เหลี่ยมอยู่ติดกับด้านหลังของลักษณะรูปร่างเกทเพื่อใช้ในการรับสายสัญญาณอินพุตที่จะถูกเชื่อมต่อเข้ากับแนนด์ โดยผู้ออกแบบสามารถเชื่อมต่อสายสัญญาณอินพุตเข้ากับส่วนรับอินพุตได้ในจำนวนไม่จำกัดแต่ต้องมีจำนวนอินพุตตั้งแต่สองอินพุตขึ้นไป ดังแสดงตัวอย่างการเชื่อมต่อสายสัญญาณอินพุตสอง สาม และสี่อินพุตในรูปที่ 4.12 โดยที่ขนาดของรูปร่างสามารถขยายได้เมื่อรูปร่างหลักถูกนำไปใช้ในการออกแบบในหน้าต่างการออกแบบ แต่จำนวนอินพุตที่ตัวจำลองการทำงานสามารถคำนวณค่าความจริงของแนนด์นั้นอยู่ที่เก้าอินพุต เนื่องจากจำนวนอินพุตของเกทชนิดนี้ที่ใช้ในการออกแบบวงจร ISCAS85 มีจำนวนไม่เกินเก้าอินพุต [24]
- ส่วนให้เอาต์พุต สายสัญญาณที่ถูกเชื่อมต่อเข้ากับส่วนให้เอาต์พุตจะมีจำนวนเพียงหนึ่งสายสัญญาณ เนื่องจากเอาต์พุตของแนนด์เกทจะมีหนึ่งเอาต์พุตเท่านั้น และสายสัญญาณดังกล่าวจะถูกกำหนดให้เป็นสายสัญญาณเอาต์พุตของแนนด์



รูปที่ 4.12 รูปร่างหลักของแนนด์ที่ส่วนรับอินพุตเป็นจำนวนสอง สาม และสี่อินพุต

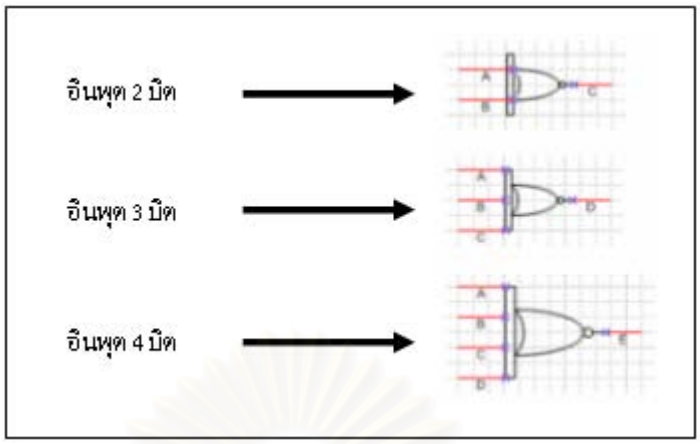
6) นอร์ (NOR) ถูกออกแบบและสร้างขึ้นเพื่อใช้เป็นสัญลักษณ์แทนนอร์เกทเพื่อคำนวณค่าความจริงของนอร์ ซึ่งนอร์เกทสร้างขึ้นโดยสร้างรูปร่างของนอร์เกทซึ่งมีลักษณะดังรูปที่ 4.13 ในหน้าต่างของการออกแบบนอร์เกทที่ออกแบบขึ้นประกอบด้วย เส้นตรงและเส้นโค้ง ซึ่งเป็นเครื่องมือพื้นฐานในการออกแบบของโปรแกรมไมโครซอฟท์ วิสิโอ จากนั้นจึงลากนอร์เกทที่ได้ออกแบบขึ้นไปปล่อยยังสเทนซิล SimulatorGate และกำหนดชื่อของรูปร่างหลักดังกล่าวให้มีชื่อว่านอร์



รูปที่ 4.13 รูปร่างหลักของนอร์

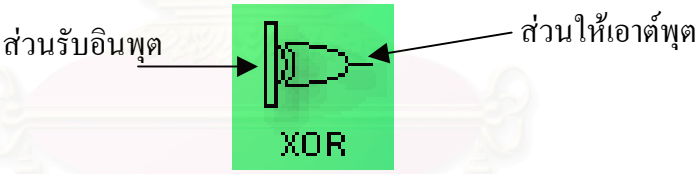
รูปร่างหลักนอร์ ประกอบด้วยสองส่วนประกอบด้วยกัน คือ

- ส่วนรับอินพุต จะถูกออกแบบให้มีรูปร่างเป็นสี่เหลี่ยมอยู่ติดกับด้านหลังของลักษณะรูปร่างเกทเพื่อใช้ในการรับสายสัญญาณอินพุตที่จะถูกเชื่อมต่อเข้ากับนอร์โดยผู้ออกแบบสามารถเชื่อมต่อสายสัญญาณอินพุตเข้ากับส่วนรับอินพุตได้ในจำนวนไม่จำกัดแต่ต้องมีจำนวนอินพุตตั้งแต่สองอินพุตขึ้นไป ดังแสดงตัวอย่างการเชื่อมต่อสายสัญญาณอินพุตสอง สาม และสี่อินพุตในรูปที่ 4.14 โดยที่ขนาดของรูปร่างสามารถขยายได้เมื่อรูปร่างหลักถูกนำไปใช้ในการออกแบบในหน้าต่างการออกแบบ แต่จำนวนอินพุตที่ตัวจำลองการทำงานสามารถคำนวณค่าความจริงของนอร์นั้น อยู่ที่เก้าอินพุต เนื่องจากจำนวนอินพุตของเกทชนิดนี้ที่ใช้ออกแบบวงจร ISCAS85 มีจำนวนไม่เกินเก้าอินพุต [24]
- ส่วนให้เอาต์พุต สายสัญญาณที่ถูกเชื่อมต่อเข้ากับส่วนให้เอาต์พุตจะมีจำนวนเพียงหนึ่งสายสัญญาณ เนื่องจากเอาต์พุตของนอร์เกทจะมีหนึ่งเอาต์พุตเท่านั้น และสายสัญญาณดังกล่าวจะถูกกำหนดให้เป็นสายสัญญาณเอาต์พุตของนอร์



รูปที่ 4.14 รูปร่างหลักของนอร์ที่ส่วนรับอินพุตเป็นจำนวนสอง สาม และสี่อินพุต

7) ออร์เฉพาะ (XOR) ถูกออกแบบและสร้างขึ้นเพื่อใช้เป็นสัญลักษณ์แทนออร์เฉพาะเกทเพื่อคำนวณค่าความจริงของออร์เฉพาะ ซึ่งออร์เฉพาะเกทสร้างขึ้นโดยสร้างรูปร่างของออร์เฉพาะเกทซึ่งมีลักษณะดังรูปที่ 4.15 ในหน้าต่างของการออกแบบออร์เฉพาะเกทที่ออกแบบขึ้นประกอบด้วย เส้นตรงและเส้นโค้ง ซึ่งเป็นเครื่องมือพื้นฐานในการออกแบบของโปรแกรมไมโครซอฟท์ วิสิโอ จากนั้นจึงลากออร์เฉพาะเกทที่ได้ออกแบบขึ้นไปปล่อยยังสเทนซิล SimulatorGate และกำหนดชื่อของรูปร่างหลักดังกล่าวให้มีชื่อว่าออร์เฉพาะ



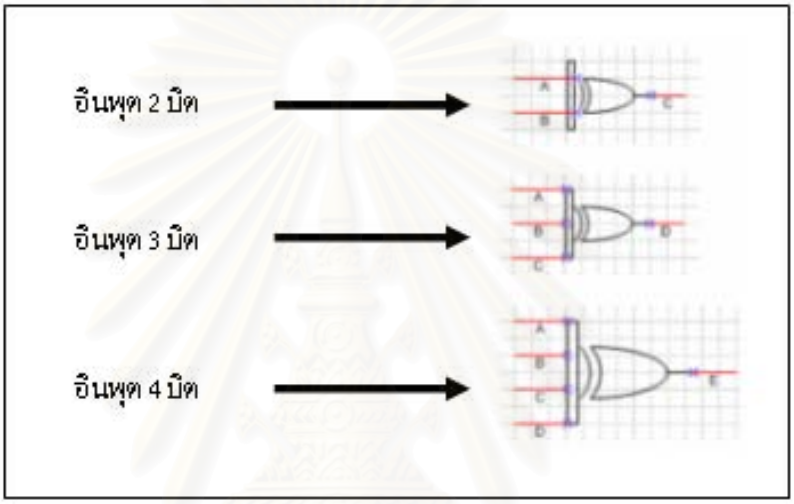
รูปที่ 4.15 รูปร่างหลักของออร์เฉพาะ

รูปร่างหลักออร์เฉพาะ ประกอบด้วยสองส่วนประกอบด้วยกัน คือ

- ส่วนรับอินพุต จะถูกออกแบบให้มีรูปร่างเป็นสี่เหลี่ยมอยู่ติดกับด้านหลังของลักษณะรูปร่างเกทเพื่อใช้ในการรับสายสัญญาณอินพุตที่จะถูกเชื่อมต่อเข้ากับออร์เฉพาะ โดยผู้ออกแบบสามารถเชื่อมต่อสายสัญญาณอินพุตเข้ากับส่วนรับอินพุตได้ในจำนวนไม่จำกัดแต่ต้องมีจำนวนอินพุตตั้งแต่สองอินพุตขึ้นไป ดังแสดงตัวอย่างการเชื่อมต่อสายสัญญาณอินพุตสอง สาม และสี่อินพุตในรูปที่ 4.16 โดยที่ขนาดของรูปร่างสามารถขยายได้เมื่อรูปร่างหลักถูกนำไปใช้ในการออกแบบในหน้าต่างการออกแบบ แต่จำนวนอินพุตที่ตัวจำลองการทำงานสามารถคำนวณค่าความจริงของออร์

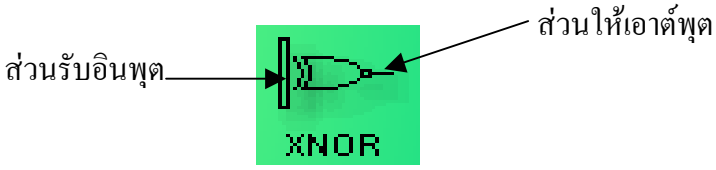
เฉพาะนั้น อยู่ที่เก้าอินพุต เนื่องจากจำนวนอินพุตของเกทชนิดนี้ที่ใช้ออกแบบวงจร ISCAS85 มีจำนวนไม่เกินเก้าอินพุต [24]

- ส่วนให้เอาต์พุต สายสัญญาณที่ถูกเชื่อมต่อเข้ากับส่วนให้เอาต์พุตจะมีจำนวนเพียงหนึ่งสายสัญญาณ เนื่องจากเอาต์พุตของออร์เฉพาะเกทจะมีหนึ่งเอาต์พุตเท่านั้น และสายสัญญาณดังกล่าวจะถูกกำหนดให้เป็นสายสัญญาณเอาต์พุตของออร์เฉพาะ



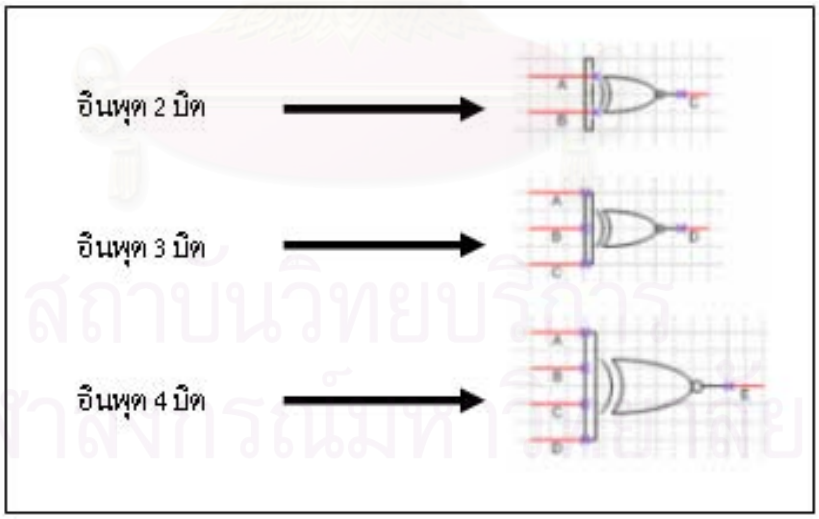
รูปที่ 4.16 รูปร่างหลักของออร์เฉพาะที่ส่วนรับอินพุตเป็นจำนวนสอง สาม และสี่อินพุต

8) ออร์ไม่เฉพาะ (XNOR) ถูกออกแบบและสร้างขึ้นเพื่อใช้เป็นสัญลักษณ์แทนออร์ไม่เฉพาะเกทเพื่อคำนวณค่าความจริงของออร์ไม่เฉพาะ ซึ่งออร์ไม่เฉพาะเกทสร้างขึ้นโดยสร้างรูปร่างของออร์ไม่เฉพาะ เกทซึ่งมีลักษณะดังรูปที่ 4.17 ในหน้าต่างของการออกแบบออร์ไม่เฉพาะเกทที่ออกแบบขึ้นประกอบด้วยเส้นตรงและเส้นโค้ง ซึ่งเป็นเครื่องมือพื้นฐานในการออกแบบของโปรแกรมไมโครซอฟท์ วิสิโอ จากนั้นจึงลากออร์ไม่เฉพาะเกทที่ได้ออกแบบขึ้นไปปล่อยยังสเทนซิล SimulatorGate และกำหนดชื่อของรูปร่างหลักดังกล่าวให้มีชื่อว่าออร์ไม่เฉพาะ รูปร่างหลักออร์ไม่เฉพาะ ประกอบด้วยสองส่วนประกอบด้วยกัน คือ



รูปที่ 4.17 รูปร่างหลักของออร์ไม่เฉพาะ

- ส่วนรับอินพุต จะถูกออกแบบให้มีรูปร่างเป็นสี่เหลี่ยมอยู่ติดกับด้านหลังของลักษณะรูปร่างเกทเพื่อใช้ในการรับสายสัญญาณอินพุตที่จะถูกเชื่อมต่อเข้ากับออร์ไม่เฉพาะ โดยผู้ออกแบบสามารถเชื่อมต่อสายสัญญาณอินพุตเข้ากับส่วนรับอินพุตได้ในจำนวนไม่จำกัดแต่ต้องมีจำนวนอินพุตตั้งแต่สองอินพุตขึ้นไป ดังแสดงตัวอย่างการเชื่อมต่อสายสัญญาณอินพุตสอง สาม และสี่อินพุตในรูปที่ 4.18 โดยที่ขนาดของรูปร่างสามารถขยายได้เมื่อรูปร่างหลักถูกนำไปใช้ในการออกแบบในหน้าต่างการออกแบบ แต่จำนวนอินพุตที่ตัวจำลองการทำงานสามารถคำนวณค่าความจริงของออร์ไม่เฉพาะนั้น อยู่ที่เก้าอินพุต เนื่องจากจำนวนอินพุตของเกทชนิดนี้ที่ใช้ในการออกแบบวงจร ISCAS85 มีจำนวนไม่เกินเก้าอินพุต [24]

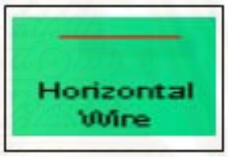


รูปที่ 4.18 รูปร่างหลักของออร์ไม่เฉพาะ ที่ส่วนรับอินพุตเป็นจำนวนสอง สาม และสี่อินพุต

- ส่วนให้เอาต์พุต สายสัญญาณที่ถูกเชื่อมต่อเข้ากับส่วนให้เอาต์พุตจะมีจำนวนเพียงหนึ่งสายสัญญาณ เนื่องจากเอาต์พุตของออร์ไม่เฉพาะเกทจะ

มีหนึ่งเอาต์พุตเท่านั้น และสายสัญญาณดังกล่าวจะถูกกำหนดให้เป็นสายสัญญาณเอาต์พุตของออร์ไม่เฉพาะ

9) สายสัญญาณในแนวนอน (Horizontal Wire) ถูกออกแบบและสร้างขึ้นเพื่อใช้เป็นสัญลักษณ์แทนสายสัญญาณในแนวนอนเพื่อเชื่อมต่อระหว่างสายสัญญาณกับสายสัญญาณ สายสัญญาณกับจุดเชื่อมต่อ สายสัญญาณกับส่วนรับอินพุตหรือสายสัญญาณกับส่วนให้เอาต์พุต ซึ่งสายสัญญาณในแนวนอนสร้างขึ้นโดยสร้างรูปร่างของสายสัญญาณในแนวนอน ซึ่งมีลักษณะดังรูปที่ 4.19 ในหน้าต่างของการออกแบบสายสัญญาณในแนวนอน ที่ออกแบบขึ้นประกอบด้วย เส้นตรง ซึ่งเป็นเครื่องมือพื้นฐานในการออกแบบของโปรแกรมไมโครซอฟท์ วิสิโอจากนั้นจึงลากสายสัญญาณในแนวนอน ที่ได้ออกแบบขึ้นไปปล่อยยังสเทนซิล SimulatorGate และกำหนดชื่อของรูปร่างหลักดังกล่าวให้มีชื่อว่าสายสัญญาณในแนวนอน



รูปที่ 4.19 รูปร่างหลักของสายสัญญาณในแนวนอน

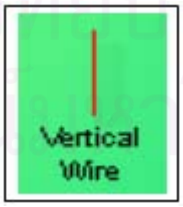
รูปร่างหลักของสายสัญญาณในแนวนอน ประกอบด้วยคุณสมบัติที่ถูกออกแบบขึ้นสี่คุณสมบัติด้วยกันคือ

- Prop.Name ทำหน้าที่เก็บชื่อของสายสัญญาณเมื่อรูปร่างหลักของ สายสัญญาณในแนวนอนถูกลากมาปล่อยและมีหน้าต่างโต้ตอบให้ผู้ออกแบบฮาร์ดแวร์กำหนดชื่อสายสัญญาณ
- Prop.Wire_Value ทำหน้าที่เก็บค่าความจริงเริ่มต้นของสายสัญญาณโดยค่าความจริงเริ่มต้นที่จะถูกจัดเก็บจะมีสองชนิด ได้แก่ FALSE เมื่อรูปร่างที่นำมาใช้เชื่อมต่อเข้ากับเกทในส่วนรับอินพุตเป็นอินพุตหลัก และค่าความจริงเริ่มต้นจะถูกจัดเก็บเป็น No Formula เมื่อรูปร่างที่นำมาใช้เชื่อมต่อเข้ากับเกทในส่วนให้เอาต์พุตเป็นเอาต์พุตหลักหรือเอาต์พุตใดๆ

- Prop.Primary_Input ทำหน้าที่เก็บค่า TRUE เมื่อรูปร่างที่นำมาใช้เชื่อมต่อเข้ากับเกทในส่วนรับอินพุตเป็นอินพุตหลัก และเก็บค่า FALSE เมื่อรูปร่างที่นำมาใช้เชื่อมต่อเข้ากับเกทในส่วนรับอินพุตเป็นอินพุตใดๆ หรือรูปร่างที่นำมาใช้เชื่อมต่อเข้ากับเกทในส่วนให้เอาต์พุต

- Prop.Primary_Output ทำหน้าที่เก็บค่า TRUE เมื่อรูปร่างที่นำมาใช้เชื่อมต่อเข้ากับเกทในส่วนให้เอาต์พุตเป็นเอาต์พุตหลัก และเก็บค่า FALSE เมื่อรูปร่างที่นำมาใช้เชื่อมต่อเข้ากับเกทในส่วนให้เอาต์พุตเป็นเอาต์พุตใดๆ หรือรูปร่างที่นำมาใช้เชื่อมต่อเข้ากับเกทในส่วนรับอินพุต

10) สายสัญญาณแนวตั้ง (Vertical Wire) ถูกออกแบบและสร้างขึ้นเพื่อใช้เป็นสัญลักษณ์แทนสายสัญญาณในแนวตั้งเพื่อเชื่อมต่อระหว่างสายสัญญาณกับสายสัญญาณ สายสัญญาณกับจุดเชื่อมต่อ สายสัญญาณกับส่วนรับอินพุต หรือสายสัญญาณกับส่วนให้เอาต์พุต ซึ่งสายสัญญาณแนวตั้งสร้างขึ้นโดยสร้างรูปร่างของสายสัญญาณแนวตั้ง ซึ่งมีลักษณะดังรูปที่ 4.20 ในหน้าต่างของการออกแบบสายสัญญาณแนวตั้งที่ออกแบบขึ้นประกอบด้วย เส้นตรง ซึ่งเป็นเครื่องมือพื้นฐานในการออกแบบของโปรแกรมไมโครซอฟท์ จากนั้นจึงลากสายสัญญาณแนวตั้งที่ได้ออกแบบขึ้นไปปล่อยยังสเทนซิล SimulatorGate และกำหนดชื่อของรูปร่างหลักดังกล่าวให้มีชื่อว่าสายสัญญาณแนวตั้ง รูปร่างหลักของสายสัญญาณแนวตั้งประกอบด้วยคุณสมบัติที่ถูกออกแบบขึ้นสี่คุณสมบัติ ซึ่งชื่อและหน้าที่เหมือนกับรูปร่างหลักของสายสัญญาณแนวนอน

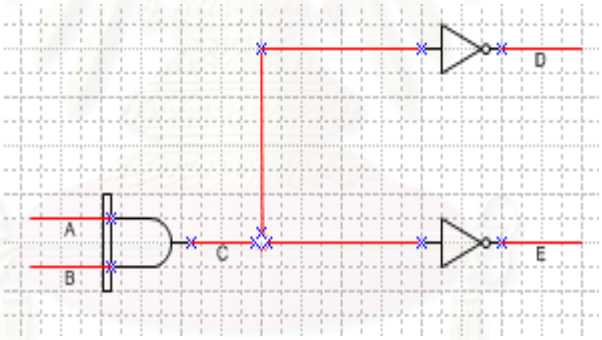


รูปที่ 4.20 รูปร่างหลักของสายสัญญาณแนวตั้ง

11) จุดเชื่อมต่อ (Connector) ถูกออกแบบและสร้างขึ้นเพื่อใช้เป็นสัญลักษณ์แทนจุดเชื่อมต่อระหว่างสายสัญญาณกับสายสัญญาณที่เป็นสายสัญญาณ

เดียวกันแต่ถูกแยกไปเป็นอินพุตของอีกเกทหนึ่ง ซึ่งจุดเชื่อมต่อ สร้างขึ้นโดยสร้างรูปร่างของจุดเชื่อมต่อ ซึ่งมีลักษณะดังรูปที่ 4.21 ในหน้าต่างของการออกแบบจุดเชื่อมต่อที่ออกแบบขึ้นประกอบด้วย เส้นโค้ง ซึ่งเป็นเครื่องมือพื้นฐานในการออกแบบของโปรแกรมไมโครซอฟท์ จากนั้นจึงลากจุดเชื่อมต่อที่ได้ออกแบบขึ้นไปปล่อยยังสเทนซิล SimulatorGate และกำหนดชื่อของรูปร่างหลักดังกล่าวให้มีชื่อว่าจุดเชื่อมต่อ และตัวอย่างแสดงการใช้รูปร่างจุดเชื่อมต่อ ดังแสดงในรูปที่ 4.22

รูปที่ 4.21 รูปร่างหลักของจุดเชื่อมต่อ



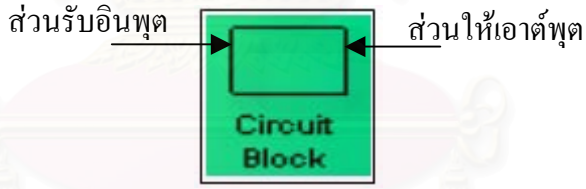
รูปที่ 4.22 ตัวอย่างการใช้รูปร่างจุดเชื่อมต่อในการออกแบบวงจร

รูปร่างหลักของจุดเชื่อมต่อประกอบด้วยคุณสมบัติที่ถูกออกแบบขึ้นสองคุณสมบัติด้วยกันคือ

- Prop.Name ทำหน้าที่เก็บชื่อของสายสัญญาณเมื่อรูปร่างของสายสัญญาณแนวตั้งหรือสายสัญญาณแนวนอน ถูกลากมาปล่อยให้จุดปลายเชื่อมเข้ากับจุดเชื่อมต่อ และจะถ่ายโอนชื่อสายสัญญาณให้กับรูปร่างของสายสัญญาณแนวตั้งหรือสายสัญญาณแนวนอน ใดๆ ที่ถูกลากมาปล่อยโดยที่จุดกำเนิดของสายสัญญาณแนวตั้งหรือสายสัญญาณแนวนอน ถูกเชื่อมเข้ากับจุดเชื่อมต่อ

- Prop.Wire_Value ทำหน้าที่เก็บค่าความจริงของสายสัญญาณเมื่อรูปร่างของสายสัญญาณแนวตั้งหรือสายสัญญาณแนวนอนถูกลากมาปล่อยให้จุดปลายเชื่อมเข้ากับจุดเชื่อมต่อและจะถ่ายโอนค่าความจริงของสายสัญญาณให้กับรูปร่างของสายสัญญาณแนวตั้งหรือสายสัญญาณแนวนอนใดๆ ที่ถูกลากมาปล่อยโดยที่จุดกำเนิดของสายสัญญาณแนวตั้งหรือสายสัญญาณแนวนอนถูกเชื่อมเข้ากับจุดเชื่อมต่อ

12) วงจรบล็อก (Circuit Block) ถูกออกแบบและสร้างขึ้นเพื่อใช้เป็นสัญลักษณ์แทนวงจรบล็อกซึ่งสนับสนุนคุณสมบัติการนำกลับมาใช้ของแนวคิดเชิงวัตถุ เพื่อให้ผู้ออกแบบฮาร์ดแวร์สามารถนำวงจรที่ได้ออกแบบไว้แล้วกลับมาใช้อีก ซึ่งวงจรบล็อก สร้างขึ้นโดยสร้างรูปร่างของวงจรบล็อก ซึ่งมีลักษณะดังรูปที่ 4.23 ในหน้าต่างของการออกแบบวงจรบล็อกที่ออกแบบขึ้นประกอบด้วย เส้นตรง ซึ่งเป็นเครื่องมือพื้นฐานในการออกแบบของโปรแกรมไมโครซอฟท์จากนั้นจึงลากวงจรบล็อกที่ได้ออกแบบขึ้นไปปล่อยยังสเทนซิล SimulatorGate และกำหนดชื่อของรูปร่างหลักดังกล่าวให้มีชื่อว่าวงจรบล็อก



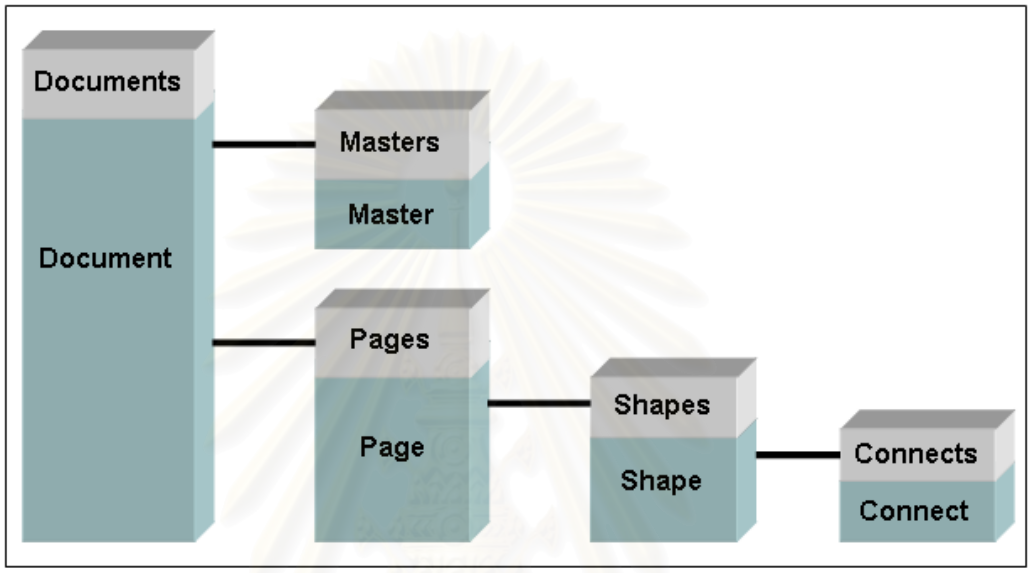
รูปที่ 4.23 รูปร่างหลักของวงจรบล็อก

รูปร่างหลักวงจรบล็อก ประกอบด้วยสองส่วนประกอบด้วยกัน คือ

- ส่วนรับอินพุต เพื่อใช้ในการรับสายสัญญาณอินพุตที่จะถูกเชื่อมต่อเข้ากับวงจรบล็อก โดยผู้ออกแบบต้องเชื่อมต่อสายสัญญาณอินพุตเข้ากับส่วนรับอินพุตในจำนวนที่เท่ากับวงจรบล็อกที่นำกลับมาใช้
- ส่วนให้เอาต์พุต สายสัญญาณที่ถูกเชื่อมต่อเข้ากับส่วนให้เอาต์พุตจะต้องมีจำนวนเท่ากับจำนวนสายสัญญาณเอาต์พุตของวงจรบล็อกที่นำมาใช้

4.1.2 การออกแบบโปรแกรมสำหรับส่วนออกแบบวงจรตรรกะเชิงผสมแบบกราฟิก

เนื้อหาในส่วนนี้จะอธิบายถึงการออกแบบทางด้านโปรแกรมในโปรแกรมไมโครซอฟท์ วิสิโอ ในส่วนของการออกแบบวงจรตรรกะเชิงผสมแบบกราฟิก โดยใช้ภาษาวิชวลเบสิกสำหรับแอพลิเคชัน ซึ่งสนับสนุนโปรแกรมไมโครซอฟท์ วิสิโออยู่ โปรแกรมไมโครซอฟท์ วิสิโอมีโครงสร้างซึ่งสนับสนุนการเชื่อมต่อระหว่างส่วนออกแบบทางกราฟิกกับการออกแบบโปรแกรม [22] โดยมีลักษณะดังแสดงในรูปที่ 4.24

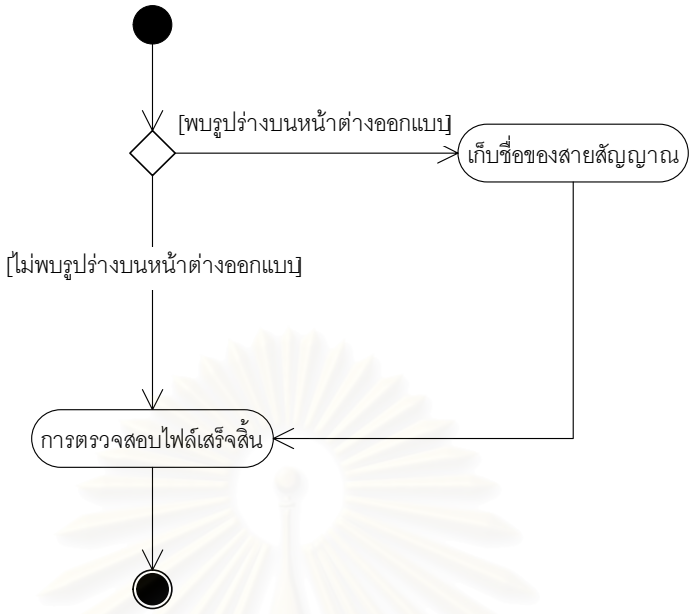


รูปที่ 4.24 โครงสร้างการเชื่อมต่อระหว่างส่วนกราฟิกและโปรแกรมของไมโครซอฟท์ วิสิโอ

โดยปกติแล้วเมื่อโปรแกรมไมโครซอฟท์ วิสิโอถูกเรียกใช้งานสำหรับการออกแบบใดๆ โปรแกรมดังกล่าวจะอ้างถึงวัตถุที่หนึ่งซึ่งมีชื่อว่า Application ให้โดยอัตโนมัติ หลังจากนั้นโปรแกรมดังกล่าวจะสร้างวัตถุที่มีชื่อว่า ThisDocument ซึ่งแสดงถึง คุณสมบัติ และฟังก์ชันที่เกี่ยวข้องกับเหตุการณ์ (Events) ที่มีความเกี่ยวข้องกับงานที่ออกแบบบนหน้าต่างของการออกแบบ โดยปริยาย ซึ่งในวัตถุ ThisDocument นี้จะประกอบด้วยวัตถุ Master คือ รูปร่างหลักที่ผู้ออกแบบฮาร์ดแวร์สามารถนำมาใช้ในการออกแบบและอ้างอิงถึงรูปร่างหลักเมื่อประมวลผลการทำงานของส่วนของโปรแกรมในโปรแกรมไมโครซอฟท์ วิสิโอ และวัตถุ Page คือ หน้าต่างออกแบบทางกราฟิกที่กำลังใช้ในการออกแบบ และใช้ในการอ้างอิงถึงหน้าต่างที่กำลังใช้ในการออกแบบเมื่อประมวลผลการทำงานของส่วนของโปรแกรมในโปรแกรมไมโครซอฟท์ วิสิโอ และเมื่อมีการออกแบบภายในหน้าต่างของการออกแบบ วัตถุชื่อ Shape คือ รูปร่างที่ถูกผู้ออกแบบฮาร์ดแวร์ลากจากรูปร่างหลักมาปล่อยยังหน้าต่างของการออกแบบ และใช้ในการอ้างอิงถึงรูปร่างที่กำลังใช้ในการออกแบบเมื่อประมวลผลการทำงานของส่วนของโปรแกรมในโปรแกรมไมโครซอฟท์ วิสิโอ เมื่อมีการเชื่อมต่อกันของรูปร่าง วัตถุชื่อ Connect จะถูกสร้างขึ้น และใช้ในการอ้างอิงถึงการเชื่อมต่อระหว่างรูปร่างที่กำลัง

พิจารณาเมื่อประมวลผลการทำงานของส่วนของโปรแกรมในโปรแกรมไมโครซอฟท์ วิสิโอ ซึ่งวัตถุแต่ละชนิดโปรแกรมไมโครซอฟท์ วิสิโอได้กำหนดพฤติกรรมเพื่อใช้สำหรับออกแบบด้วยส่วนของโปรแกรม ในส่วนของการออกแบบโปรแกรม ผู้ออกแบบฮาร์ดแวร์สามารถสร้างคุณสมบัติและฟังก์ชันของเหตุการณ์ใดๆ เพิ่มเติมเพื่อเพิ่มประสิทธิภาพของการออกแบบให้ดีขึ้นได้ ดังนั้นในวิทยานิพนธ์นี้ในส่วนของการออกแบบวงจรตรรกะเชิงผสมแบบกราฟิกนอกจากที่ได้ออกแบบสเทนซิลและรูปร่างหลักแล้ว ยังได้มีการออกแบบเพิ่มเติมโปรแกรมในวัตถุ ThisDocument อีกด้วย โดยการออกแบบโปรแกรมในวัตถุ ThisDocument ประกอบไปด้วยสี่ฟังก์ชันซึ่งรองรับกับเหตุการณ์ที่จะเกิดขึ้นจากการออกแบบของผู้ออกแบบฮาร์ดแวร์ และสองส่วนของโปรแกรม ดังนี้

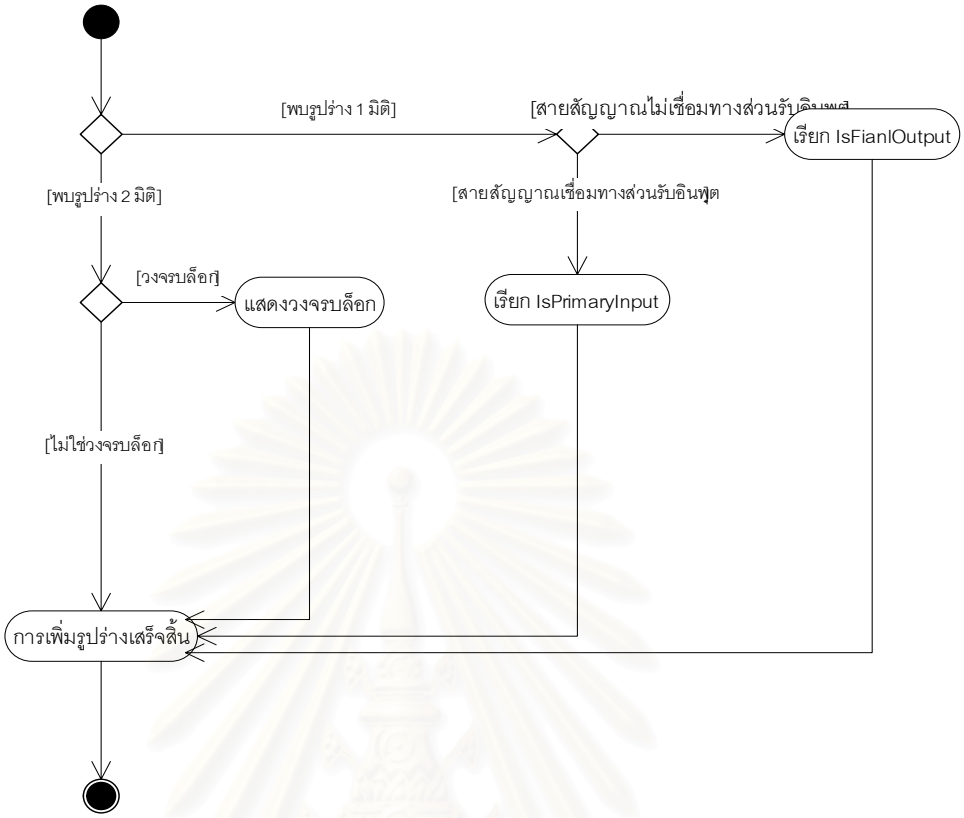
4.1.2.1 Document_DocumentOpened(ByVal doc As IVDocument) เป็นฟังก์ชันของเหตุการณ์ที่จะทำงาน เมื่อผู้ออกแบบฮาร์ดแวร์ได้เรียกส่วนออกแบบวงจรตรรกะเชิงผสมแบบกราฟิกทำงาน ไม่ว่าจะเป็นการเรียกใหม่หรือเรียกจากไฟล์ที่มีการออกแบบค้างไว้ ฟังก์ชันของเหตุการณ์นี้ถูกออกแบบให้มีการทำงานดังแสดงในรูปที่ 4.25 คือ เมื่อส่วนออกแบบวงจรตรรกะเชิงผสมแบบกราฟิกถูกเรียกใช้ฟังก์ชันของเหตุการณ์ดังกล่าวถูกเรียกให้ทำงาน จะตรวจสอบว่าภายในหน้าต่างการออกแบบมีการออกแบบค้างไว้หรือไม่ ถ้ามี จะแสดงถึงการเรียกส่วนออกแบบวงจรตรรกะเชิงผสมจากไฟล์ที่ได้ออกแบบค้างไว้ โดยจะสังเกตจากรูปร่างที่ปรากฏอยู่ในหน้าต่างของการออกแบบ แต่ถ้าไม่มี จะแสดงถึงการเรียกส่วนออกแบบวงจรตรรกะเชิงผสมเป็นการเรียกใหม่ สังเกตโดยจะไม่มีรูปร่างใดปรากฏอยู่บนหน้าต่างของการออกแบบ ซึ่งการสังเกตว่ามีรูปร่างหลักหรือไม่นั้น โดยนับจำนวนของรูปร่างบนหน้าต่างของการออกแบบโดยเรียกพฤติกรรมชื่อ Count ของวัตถุ Page เพื่อนับจำนวนของรูปร่าง ถ้าฟังก์ชันของเหตุการณ์ตรวจไม่พบรูปร่างใดๆ บนหน้าต่างการออกแบบ ฟังก์ชันของเหตุการณ์นี้จะไม่ทำสิ่งใด แต่ถ้าตรวจพบรูปร่างบนหน้าต่างของการออกแบบ จะตรวจสอบว่าเป็นรูปร่างที่เกิดจากรูปร่างหลักของสายสัญญาณแนวนอนหรือสายสัญญาณแนวตั้งหรือไม่ ซึ่งวิธีการตรวจสอบ คือ ตรวจสอบชื่อของรูปร่างหลัก โดยเรียกพฤติกรรมชื่อ name จากวัตถุ Shape ของรูปร่าง ถ้าการตรวจสอบปรากฏว่าไม่ใช่รูปร่างที่เกิดจากรูปร่างหลักทั้งสองนี้ก็จะไม่ทำสิ่งใด แต่ถ้าเกิดจากรูปร่างหลักทั้งสองชนิดนี้หรือชนิดใดชนิดหนึ่งฟังก์ชันของเหตุการณ์นี้จะเก็บคุณสมบัติที่ถูกสร้างขึ้นที่มีชื่อว่า



รูปที่ 4.25 การทำงานของฟังก์ชันของเหตุการณ์ Document_DocumentOpened

Prop.Name โดยคุณสมบัตินี้คือ ชื่อสายสัญญาณของรูปร่าง ไว้ในโครงสร้างข้อมูลชนิดคอลเล็กชัน (Collection) โดยก่อนที่จะเก็บจะมีการตรวจสอบข้อมูลในคอลเล็กชันก่อนว่ามีชื่อที่ต้องการจะเก็บอยู่แล้วหรือไม่ ถ้ามี ก็จะไม่เก็บชื่อดังกล่าวลงไป แต่ถ้าไม่มี ก็จะเพิ่มชื่อสายสัญญาณนั้นลงไปยังคอลเล็กชัน โดยวัตถุประสงค์ของการเก็บข้อมูลไว้ในคอลเล็กชันนี้ คือ เพื่อป้องกันการกำหนดชื่อซ้ำให้กับสายสัญญาณในการออกแบบวงจร

4.1.2.2 Document_ShapeAdded(ByVal Shape As IVShape) เป็นฟังก์ชันของเหตุการณ์ที่จะถูกเรียกใช้ทำงานเมื่อมีการลากรูปร่างหลักมาปล่อยลงยังหน้าต่างของการออกแบบ หรือถูกเรียกว่า การเพิ่มรูปร่างหลัก ฟังก์ชันของเหตุการณ์นี้ถูกออกแบบให้มีการทำงานดังแสดงในรูปที่ 4.26 คือ เมื่อมีรูปร่างที่ถูกลากมาปล่อยบนหน้าต่างการออกแบบ จะถูกตรวจสอบว่า รูปร่างดังกล่าวเป็นรูปร่างแบบหนึ่งมิติหรือสองมิติ โดยเรียกพฤติกรรม oneD จากวัตถุ Shape ของรูปร่างที่ได้ออกแบบไว้ ซึ่งถ้าเป็นรูปร่างหนึ่งมิติจะแสดงถึงรูปร่างหลักของสายสัญญาณแนวนอนหรือสายสัญญาณแนวตั้งแต่ถ้าเป็นรูปร่างสองมิติ จะแสดงถึงรูปร่างหลักของแอนด์ ออร์ บัฟเฟอร์ ตัวผกผัน แนนด์ นอร์ ออร์เฉพาะ ออร์ไม่เฉพาะ จุดเชื่อมต่อ หรือวงจรบล็อก โดยถ้าเป็นรูปร่างหนึ่งมิติ ฟังก์ชันของเหตุการณ์จะตรวจสอบว่าเป็นการเชื่อมต่อกับรูปร่างอื่นหรือไม่ และจำนวนการเชื่อมต่อว่ามี



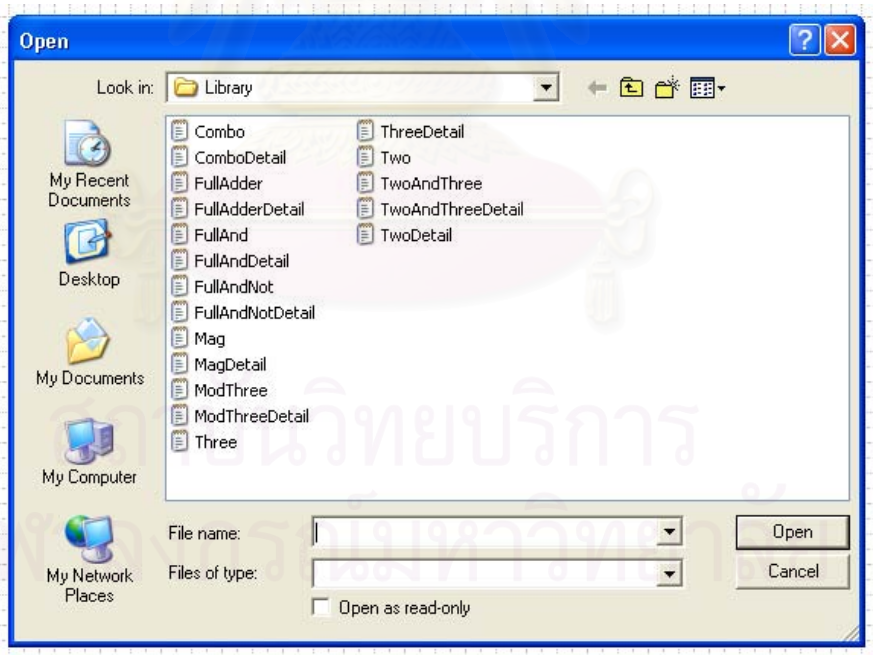
รูปที่ 4.26 การทำงานของฟังก์ชันของเหตุการณ์ Document_ShapeAdded

หนึ่งหรือสองจำนวน โดยที่ถ้ารูปร่างดังกล่าวไม่มีการเชื่อมต่อกับรูปร่างอื่น ฟังก์ชันของเหตุการณ์จะไม่ทำสิ่งใด แต่ถ้ามีการเชื่อมต่อกับรูปร่างอื่นเพียงหนึ่งจุด ฟังก์ชันของเหตุการณ์จะตรวจสอบต่อไปว่า รูปร่างหนึ่งมิติดังกล่าวซึ่งอาจเป็นรูปร่างของรูปร่างหลักสายสัญญาณแนวนอนหรือสายสัญญาณแนวตั้งนั้นมีตำแหน่งของจุดเริ่มต้นทางแกน X อยู่น้อยกว่าจุดกึ่งกลางของรูปร่างที่ถูกเชื่อมต่อที่เป็นสองมิติ โดยพิจารณาจากแนวแกน X เท่านั้น หรือไม่ โดยถ้ามีค่าน้อยกว่าจริง ฟังก์ชันของเหตุการณ์นี้จะเรียกส่วนของโปรแกรมที่มีชื่อว่า IsPrimary(Shape) พร้อมกับส่งรูปร่างหนึ่งมิติดังกล่าวเป็นอาร์กิวเมนต์ (Argument) ไปด้วย แต่ถ้าตำแหน่งของจุดเริ่มต้นทางแกน X ของรูปร่างหนึ่งมิติ มีค่ามากกว่าจุดกึ่งกลางตามแนวแกน X ของรูปร่างที่ถูกเชื่อมต่อเป็นสองมิติ ฟังก์ชันของเหตุการณ์นี้จะเรียกส่วนของโปรแกรม IsFinalOutput(Shape) พร้อมกับส่งรูปร่างหนึ่งมิติดังกล่าวเป็นอาร์กิวเมนต์ไปด้วย แต่ถ้าการตรวจสอบรูปร่างที่ถูกนำมาปล่อยในหน้าต่างของการออกแบบเป็นรูปร่างสองมิติ ฟังก์ชันของเหตุการณ์นี้จะตรวจสอบว่าเป็นรูปร่างหลักของ Circuit Block หรือไม่ ถ้าไม่ใช่ฟังก์ชันของเหตุการณ์นี้จะไม่ทำสิ่งใด แต่ถ้า

ใช่ฟังก์ชันของเหตุการณ์นี้จะเรียกส่วนติดต่อกับผู้ออกแบบฮาร์ดแวร์ที่มีชื่อว่า UserForm6 ดังแสดงในรูปที่ 4.27 ขึ้นมา

รูปที่ 4.27 ส่วนติดต่อกับผู้ออกแบบฮาร์ดแวร์ชื่อ UserForm6

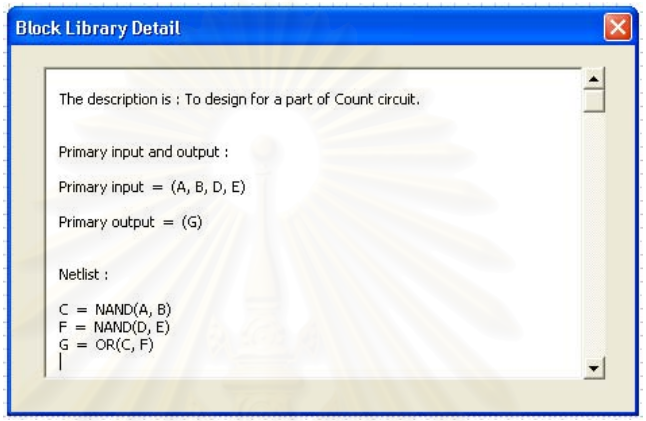
เมื่อผู้ออกแบบฮาร์ดแวร์กดปุ่ม Open File แล้วหน้าต่างติดต่อกับผู้ออกแบบฮาร์ดแวร์สำหรับเลือกไฟล์ของวงจรบล็อก ซึ่งมีนามสกุล .txt ดังแสดงในรูปที่ 4.28 จะปรากฏขึ้น



รูปที่ 4.28 หน้าต่างติดต่อกับผู้ออกแบบฮาร์ดแวร์สำหรับเลือกไฟล์ของวงจรบล็อก

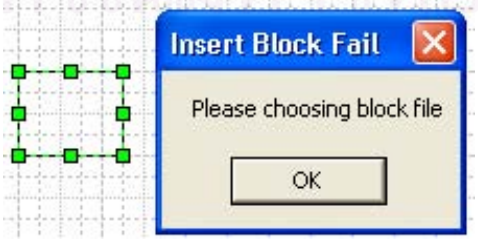
ซึ่งเมื่อผู้ออกแบบฮาร์ดแวร์เลือกไฟล์วงจรบล็อกที่ต้องการแล้วกดปุ่ม Open ฟังก์ชันของเหตุการณ์นี้จะไปอ่านไฟล์ของวงจรบล็อกที่มีชื่อตามที่ผู้ออกแบบฮาร์ดแวร์เรียกแล้วตามด้วยคำว่า Detail ซึ่งอยู่ในรูปแบบของไฟล์ข้อความ (Text

File) แสดงอินพุต เอาต์พุตหลัก พร้อมทั้งเนทลิสต์ที่ผ่านการสังเคราะห์วงจรจากตัวสังเคราะห์วงจรของวงจรบล็อกที่ผู้ออกแบบฮาร์ดแวร์เลือกแล้วแสดงออกทางหน้าต่างติดต่อกับผู้ออกแบบฮาร์ดแวร์ที่มีชื่อว่า UserForm4 ดังแสดงตัวอย่างในรูปที่ 4.29



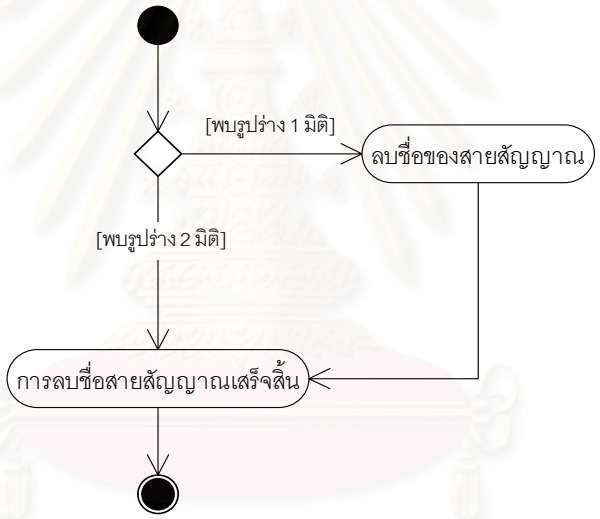
รูปที่ 4.29 หน้าต่างติดต่อกับผู้ออกแบบฮาร์ดแวร์ชื่อ UserForm4

แล้วฟังก์ชันของเหตุการณ์นี้จะนำเอาชื่อของวงจรบล็อกนำไปเขียนให้อยู่ตำแหน่งกึ่งกลางของรูปร่างของวงจรบล็อกที่ถูกนำมาวางบนหน้าต่างของการออกแบบจากหน้าต่างติดต่อกับผู้ออกแบบฮาร์ดแวร์สำหรับเลือกไฟล์ของวงจรบล็อก ถ้าผู้ออกแบบฮาร์ดแวร์กดปุ่ม Open โดยไม่ได้เลือกไฟล์วงจรบล็อกใดๆ ฟังก์ชันของเหตุการณ์นี้จะกำหนดให้เลขจำนวนจริงถูกหารด้วยเลขศูนย์ เพื่อให้เกิดข้อผิดพลาดในเหตุการณ์นี้ขึ้น เนื่องจากฟังก์ชันของเหตุการณ์นี้ถูกออกแบบให้มีการจัดการข้อผิดพลาด (Exception Handling) ที่เกิดขึ้น โดยจะไปเรียก Exception ที่มีชื่อว่า divideByZeroHandler และแสดงกล่องข้อความเตือนแก่ผู้ออกแบบฮาร์ดแวร์ ดังแสดงในรูปที่ 4.30



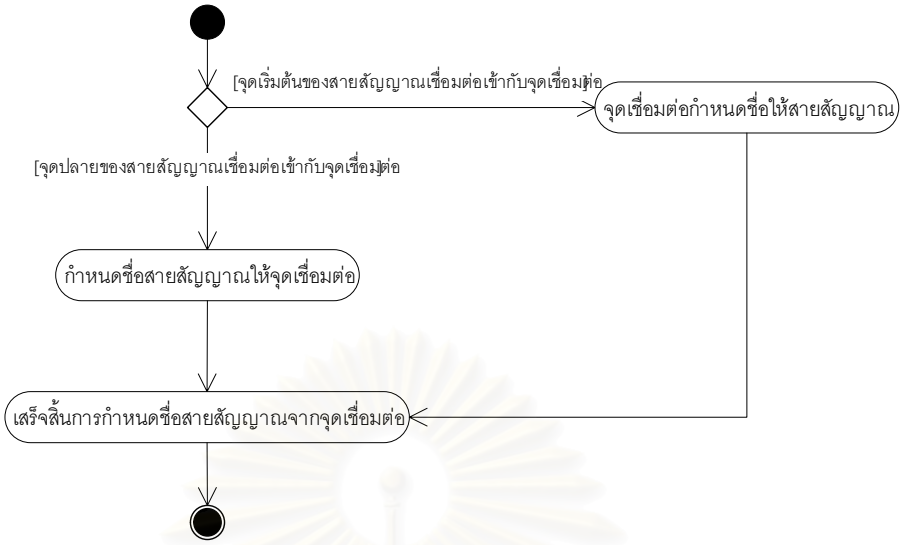
รูปที่ 4.30 กล่องข้อความเตือนกับผู้ออกแบบฮาร์ดแวร์

4.1.2.3 thePage_BeforeShapeDelete(ByVal Shape As IVShape) เป็นฟังก์ชันของเหตุการณ์ที่จะถูกเรียกใช้และทำงานเมื่อรูปร่างที่ปรากฏอยู่บนหน้าต่างของการออกแบบได้ถูกลบทิ้ง ซึ่งฟังก์ชันของเหตุการณ์นี้จะตรวจสอบว่ารูปร่างที่ถูกลบทิ้งนี้เป็นรูปร่างหนึ่งมิติหรือไม่ซึ่งก็คือรูปร่างหลักของ Horizontal Wire หรือ Vertical Wire ถ้ารูปร่างที่ปรากฏอยู่บนหน้าต่างของการออกแบบไม่ใช่รูปร่างหนึ่งมิติ ฟังก์ชันของเหตุการณ์นี้จะไม่ทำสิ่งใด แต่ถ้าเป็นรูปร่างหนึ่งมิติ ฟังก์ชันของเหตุการณ์นี้จะตรวจสอบชื่อของสายสัญญาณซึ่งเป็นคุณสมบัติหนึ่งของรูปร่างหลักสายสัญญาณแนวนอนและสายสัญญาณแนวตั้งที่ถูกสร้างขึ้น กับคอลเล็กชันที่เก็บชื่อของสายสัญญาณไว้ ซึ่งถ้าตรงกัน ให้ทำการลบชื่อสายสัญญาณดังกล่าวออกจากคอลเล็กชัน เพื่อให้ชื่อของสายสัญญาณดังกล่าวสามารถนำกลับมาใช้ได้อีกครั้งหนึ่ง ดังแสดงการทำงานของฟังก์ชันของเหตุการณ์ในรูปที่ 4.31



รูปที่ 4.31 การทำงานของฟังก์ชันของเหตุการณ์ thePage_BeforeShapeDelete

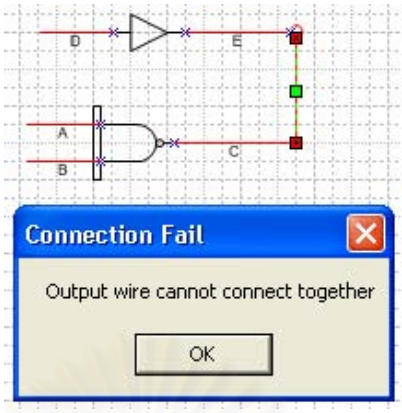
4.1.2.4 thePage_ConnectionsAdded(ByVal Connects As IVConnects) เป็นฟังก์ชันของเหตุการณ์ที่จะถูกเรียกใช้เมื่อรูปร่างของรูปร่างหลักสายสัญญาณแนวนอน สายสัญญาณแนวตั้งหรือจุดเชื่อมต่อที่ถูกนำมาออกแบบบนหน้าต่างการออกแบบมีการเชื่อมต่อกัน ซึ่งการทำงานของฟังก์ชันของเหตุการณ์นี้แสดงดังรูปที่ 4.32 โดยฟังก์ชันของเหตุการณ์นี้จะตรวจสอบการเชื่อมต่อระหว่างรูปร่างของรูปร่างหลักจุดเชื่อมต่อว่าถูกเชื่อมต่อด้วยจุดปลายหรือจุดเริ่มต้นของรูปร่างซึ่งเกิดจากรูปร่างหลักสายสัญญาณแนวตั้งหรือสายสัญญาณแนวนอน ถ้าถูกเชื่อมต่อด้วย



รูปที่ 4.32 การทำงานของฟังก์ชันของเหตุการณ์ thePage_ConnectionsAdded

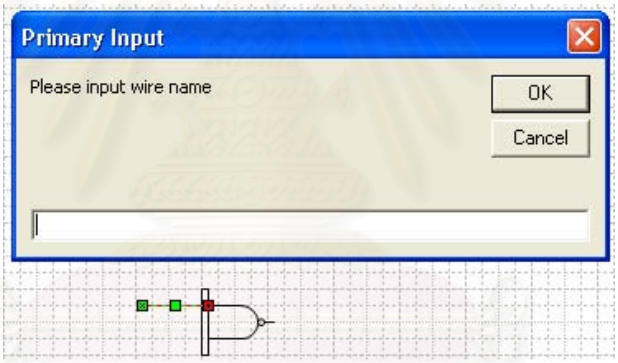
จุดปลาย ชื่อของสายสัญญาณที่มีชื่อว่า Prop.Name ซึ่งเป็นคุณสมบัติหนึ่งที่ถูกสร้างขึ้นของรูปร่างหลักสายสัญญาณแนวตั้งหรือสายสัญญาณแนวนอนจะถูกนำมาเปรียบเทียบกับ Prop.Name ซึ่งเป็นคุณสมบัติหนึ่งที่ถูกสร้างขึ้นของรูปร่างหลักจุดเชื่อมต่อ ซึ่งถ้า Prop.Name ของรูปร่างซึ่งเกิดจากรูปร่างหลักจุดเชื่อมต่อเป็นค่าปริยาย คือ 0.0000p ชื่อของสายสัญญาณจะถูกกำหนดแทนค่านี้ แต่ถ้า Prop.Name ของรูปร่างซึ่งเกิดจากรูปร่างหลักจุดเชื่อมต่อไม่ใช่ค่าปริยาย แสดงว่าสายสัญญาณสองเส้นนี้ถูกเชื่อมต่อกันแบบไม่ถูกต้องตามหลักการออกแบบวงจรตรรกะเชิงผสม ดังนั้นฟังก์ชันของเหตุการณ์นี้จะทำให้เกิดการหารเลขจำนวนเต็มด้วยค่าศูนย์ เพื่อให้เกิดความผิดพลาดในเหตุการณ์ เนื่องจากฟังก์ชันของเหตุการณ์นี้ถูกออกแบบให้มีการจัดการข้อผิดพลาดที่เกิดขึ้น โดยจะไปเรียก Exception ที่มีชื่อว่า divideByZeroHandler1 และแสดงกล่องข้อความเตือนการเชื่อมต่อสายสัญญาณที่ไม่ถูกต้องตามหลักการออกแบบวงจรตรรกะเชิงผสม ซึ่งรูปที่ 4.33 แสดงตัวอย่างการออกแบบวงจรตรรกะเชิงผสมที่ไม่ถูกต้องและแสดงกล่องข้อความเพื่อเตือน

4.1.2.5 IsPrimaryInput(ByVal shpObj As Visio.Shape) เป็นส่วนของโปรแกรมที่จะถูกเรียกใช้จากฟังก์ชันของเหตุการณ์ Document_ShapeAdded(ByVal Shape As IVShape) ทำหน้าที่รับชื่อของสายสัญญาณจากผู้ออกแบบฮาร์ดแวร์ โดยเมื่อส่วนของโปรแกรมนี้ถูกเรียกใช้จะแสดงส่วนติดต่อกับผู้ออกแบบฮาร์ดแวร์เพื่อให้



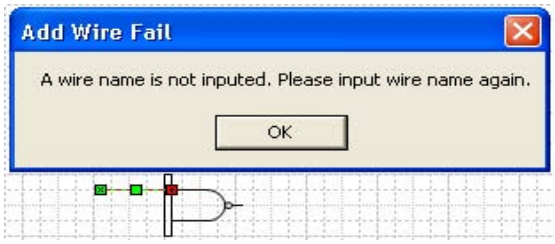
รูปที่ 4.33 ตัวอย่างการเชื่อมต่อสายสัญญาณที่ไม่ถูกต้องและกล่องข้อความแสดงข้อความเตือน

ผู้ออกแบบฮาร์ดแวร์ใส่ชื่อของสายสัญญาณที่เป็นอินพุตหลัก ดังแสดงในรูปที่ 4.34



รูปที่ 4.34 ส่วนติดต่อกับผู้ออกแบบฮาร์ดแวร์เพื่อรับชื่อสายสัญญาณอินพุตหลัก

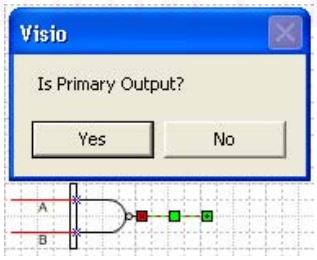
ส่วนของโปรแกรมจะตรวจสอบว่าผู้ออกแบบฮาร์ดแวร์ได้กำหนดชื่อของสายสัญญาณหรือไม่ ถ้าผู้ออกแบบฮาร์ดแวร์ไม่ได้กำหนดชื่อสายสัญญาณแล้วกดปุ่ม OK ส่วนของโปรแกรมนี้ก็จะทำให้เกิดการหารเลขจำนวนเต็มด้วยค่าศูนย์ เพื่อให้เกิดความผิดพลาดในฟังก์ชันของเหตุการณ์ เนื่องจากฟังก์ชันของเหตุการณ์นี้ถูกออกแบบให้มีการจัดการข้อผิดพลาดที่เกิดขึ้น โดยจะไปเรียก Exception ที่มีชื่อว่า divideByZeroHandler และแสดงกล่องข้อความเตือนให้ผู้ออกแบบฮาร์ดแวร์กำหนดชื่อสายสัญญาณ ซึ่งรูปที่ 4.35 แต่ถ้าผู้ออกแบบฮาร์ดแวร์ได้กำหนดชื่อให้กับสายสัญญาณแล้วส่วนของโปรแกรมจะทำการตรวจสอบชื่อของ



รูปที่ 4.35 กล่องข้อความเตือนผู้ออกแบบฮาร์ดแวร์ให้กำหนดชื่อสายสัญญาณ

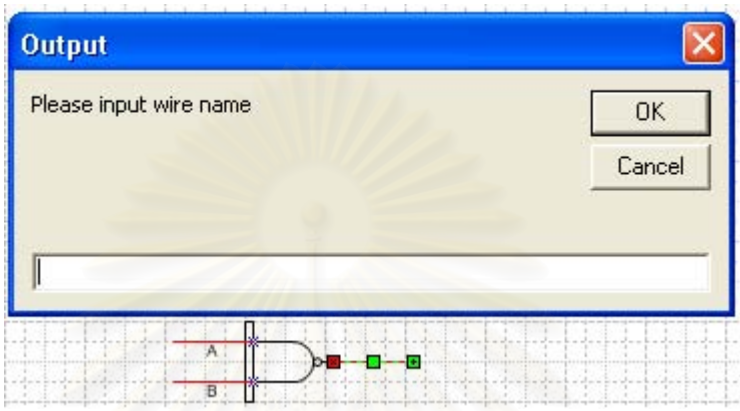
สายสัญญาณที่ถูกกำหนดว่า ได้มีการใช้ชื่อสายสัญญาณนี้หรือไม่ ถ้าปรากฏว่าเป็นชื่อซ้ำ ส่วนของโปรแกรมนี้ก็จะทำให้เกิดการหารเลขจำนวนเต็มด้วยค่าศูนย์ เพื่อให้เกิดความผิดพลาดในฟังก์ชันของเหตุการณ์ เนื่องจากฟังก์ชันของเหตุการณ์นี้ถูกออกแบบให้มีการจัดการข้อผิดพลาดที่เกิดขึ้น โดยจะไปเรียก Exception ที่มีชื่อว่า divideByZeroHandler1 และแสดงกล่องข้อความเตือนการกำหนดชื่อสายสัญญาณซ้ำ แต่ถ้าชื่อสายสัญญาณดังกล่าวไม่ใช่ชื่อซ้ำ ส่วนของโปรแกรมจะแสดงชื่อของสายสัญญาณในตำแหน่งที่ใกล้เคียงกับรูปร่างของสายสัญญาณดังกล่าว พร้อมทั้งกำหนดค่าความจริงที่เป็นจริงให้กับคุณสมบัติ Prop.PrimaryInput และค่าความจริงที่เป็นเท็จให้กับคุณสมบัติ Prop.PrimaryOutput ซึ่งคุณสมบัติทั้งสองเป็นคุณสมบัติที่ถูกออกแบบและสร้างขึ้นในรูปร่างหลักของ สายสัญญาณแนวนอนและสายสัญญาณแนวตั้ง

4.1.2.6 IsFinalOutput(ByVal shpObj As Visio.Shape) เป็นส่วนของโปรแกรมที่จะถูกเรียกใช้จากฟังก์ชันของเหตุการณ์ Document_ShapeAdded(ByVal Shape As IVShape) ทำหน้าที่รับชื่อของสายสัญญาณจากผู้ออกแบบฮาร์ดแวร์ เมื่อส่วนของโปรแกรมถูกเรียกใช้ จะแสดงหน้าต่างโต้ตอบเพื่อสอบถามผู้ออกแบบฮาร์ดแวร์ว่าสายสัญญาณเอาต์พุตที่ผู้ออกแบบฮาร์ดแวร์ออกแบบเป็นเอาต์พุตหลักหรือเอาต์พุตใดๆ ดังแสดงในรูปที่ 4.36



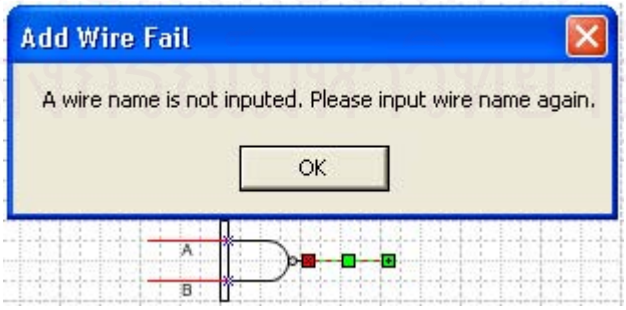
รูปที่ 4.36 หน้าต่างโต้ตอบเพื่อเลือกว่าเป็นเอาต์พุตหลักหรือเอาต์พุตใดๆ

ถ้าผู้ออกแบบฮาร์ดแวร์ลือกว่าเป็นเอาต์พุตหลัก ส่วนของโปรแกรมจะแสดงหน้าต่างโต้ตอบกับผู้ออกแบบฮาร์ดแวร์เพื่อให้ผู้ออกแบบฮาร์ดแวร์กำหนดชื่อของสายสัญญาณเอาต์พุต ดังแสดงในรูปที่ 4.37



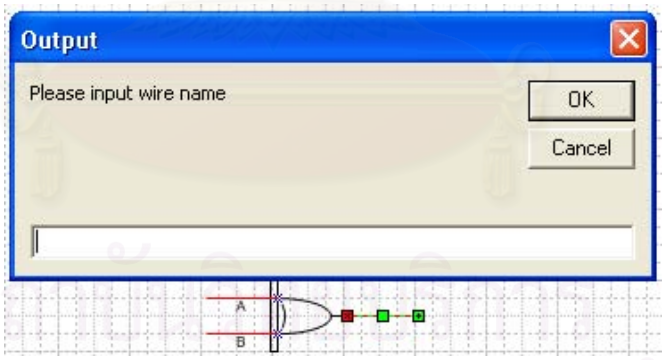
รูปที่ 4.37 หน้าต่างโต้ตอบเพื่อรับชื่อของสายสัญญาณเอาต์พุตหลัก

ส่วนของโปรแกรมจะตรวจสอบว่าผู้ออกแบบฮาร์ดแวร์ได้กำหนดชื่อของสายสัญญาณหรือไม่ ถ้าผู้ออกแบบฮาร์ดแวร์ไม่ได้กำหนดชื่อสายสัญญาณแล้วกดปุ่ม OK ส่วนของโปรแกรมนี้ก็จะทำให้เกิดการหารเลขจำนวนเต็มด้วยค่าศูนย์ ซึ่งจะทำให้เกิดความผิดพลาดในฟังก์ชันของเหตุการณ์ เนื่องจากฟังก์ชันของเหตุการณ์นี้ถูกออกแบบให้มีการจัดการข้อผิดพลาดที่เกิดขึ้น โดยจะไปเรียก Exception ที่มีชื่อว่า divideByZeroHandler และแสดงกล่องข้อความเตือนผู้ออกแบบฮาร์ดแวร์ให้กำหนดชื่อสายสัญญาณ ซึ่งรูปที่ 4.38



รูปที่ 4.38 กล่องข้อความเตือนผู้ออกแบบฮาร์ดแวร์ให้กำหนดชื่อสายสัญญาณ

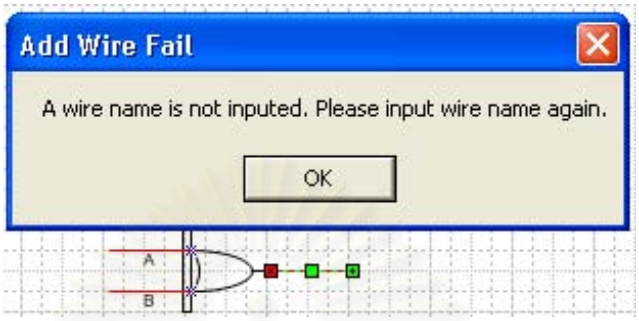
แต่ถ้าผู้ออกแบบฮาร์ดแวร์ได้กำหนดชื่อให้กับสายสัญญาณแล้ว ส่วนของโปรแกรมจะทำการตรวจสอบชื่อของสายสัญญาณที่ถูกกำหนดว่าได้มีการใช้ชื่อสายสัญญาณนี้หรือไม่ ถ้าปรากฏว่าเป็นชื่อซ้ำ ส่วนของโปรแกรมนี้ก็จะทำให้เกิดการหารเลขจำนวนเต็มด้วยค่าศูนย์ ซึ่งจะทำให้เกิดความผิดพลาดในเหตุการณ์ เนื่องจากฟังก์ชันของเหตุการณ์นี้ถูกออกแบบให้มีการจัดการข้อผิดพลาดที่เกิดขึ้น โดยจะไปเรียก Exception ที่มีชื่อว่า divideByZeroHandler2 และแสดงกล่องข้อความเตือนการกำหนดชื่อสายสัญญาณซ้ำ แต่ถ้าชื่อสายสัญญาณดังกล่าวไม่ใช่ชื่อซ้ำ ส่วนของโปรแกรมจะแสดงชื่อของสายสัญญาณในตำแหน่งที่ใกล้เคียงกับรูปร่างของสายสัญญาณดังกล่าว พร้อมทั้งกำหนดค่าความจริงเท็จให้กับคุณสมบัติ Prop.PrimaryInput และค่าความจริงที่เป็นจริงให้กับคุณสมบัติ Prop.PrimaryOutput ซึ่งคุณสมบัติทั้งสองเป็นคุณสมบัติที่ถูกออกแบบและสร้างขึ้นในรูปร่างหลักของ สายสัญญาณแนวนอนและสายสัญญาณแนวตั้ง แต่ถ้าผู้ออกแบบฮาร์ดแวร์กำหนดสายสัญญาณให้เป็นเอาต์พุตใดๆ ส่วนของโปรแกรมจะแสดงหน้าต่างโต้ตอบกับผู้ออกแบบฮาร์ดแวร์เพื่อให้ผู้ออกแบบฮาร์ดแวร์กำหนดชื่อของสายสัญญาณเอาต์พุต ดังแสดงในรูปที่ 4.39



รูปที่ 4.39 หน้าต่างโต้ตอบเพื่อรับชื่อของสายสัญญาณเอาต์พุตใดๆ

ส่วนของโปรแกรมจะตรวจสอบว่าผู้ออกแบบฮาร์ดแวร์ได้กำหนดชื่อของสายสัญญาณหรือไม่ ถ้าผู้ออกแบบฮาร์ดแวร์ไม่ได้กำหนดชื่อสายสัญญาณแล้วกดปุ่ม OK ส่วนของโปรแกรมนี้ก็จะทำให้เกิดการหารเลขจำนวนเต็มด้วยค่าศูนย์ ซึ่งจะทำให้เกิดความผิดพลาดในฟังก์ชันของเหตุการณ์ เนื่องจากฟังก์ชันของเหตุการณ์นี้ถูกออกแบบให้มีการจัดการข้อผิดพลาดที่เกิดขึ้น โดยจะไปเรียก Exception ที่มีชื่อว่า divideByZeroHandler1 และแสดงกล่องข้อความเตือนผู้ออกแบบ

ฮาร์ดแวร์ให้กำหนดชื่อสายสัญญาณ ซึ่งรูปที่ 4.40 แต่ถ้าผู้ออกแบบฮาร์ดแวร์ได้กำหนดชื่อให้กับสายสัญญาณแล้ว ส่วนของโปรแกรมจะทำการตรวจสอบชื่อของสายสัญญาณที่ถูกกำหนดว่าได้มีการใช้



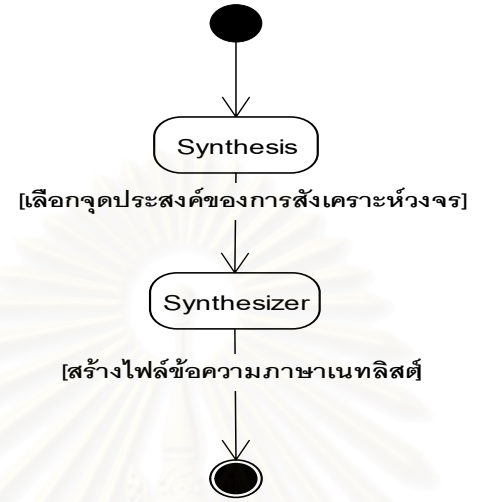
รูปที่ 4.40 กล่องข้อความเตือนให้ผู้ออกแบบฮาร์ดแวร์กำหนดชื่อสายสัญญาณ

ชื่อสายสัญญาณนี้หรือไม่ ถ้าปรากฏว่าเป็นชื่อซ้ำ ส่วนของโปรแกรมนี้ก็จะทำให้เกิดการหารเลขจำนวนเต็มด้วยค่าศูนย์ ซึ่งจะทำให้เกิดความผิดพลาดในฟังก์ชันของเหตุการณ์ เนื่องจากฟังก์ชันของเหตุการณ์นี้ถูกออกแบบให้มีการจัดการข้อผิดพลาดที่เกิดขึ้น โดยจะไปเรียก Exception ที่มีชื่อว่า divideByZeroHandler3 เพื่อแสดงกล่องข้อความเตือนการกำหนดชื่อสายสัญญาณซ้ำ แต่ถ้าชื่อสายสัญญาณดังกล่าวไม่ใช่ชื่อซ้ำ ส่วนของโปรแกรมจะแสดงชื่อของสายสัญญาณในตำแหน่งที่ใกล้เคียงกับรูปร่างของสายสัญญาณดังกล่าว พร้อมทั้งกำหนดค่าความจริงเท็จให้กับคุณสมบัติ Prop.PrimaryInput และค่าความจริงเท็จให้กับคุณสมบัติ Prop.PrimaryOutput ซึ่งคุณสมบัติทั้งสองเป็นคุณสมบัติที่ถูกออกแบบและสร้างขึ้นในรูปร่างหลักของ สายสัญญาณแนวนอนและสายสัญญาณแนวตั้ง

4.2 **เครื่องมือสังเคราะห์วงจร** (Synthesizer)

เครื่องมือสังเคราะห์วงจรถูกออกแบบและสร้างขึ้นภายในโมดูล (Module) ของโปรแกรมไมโครซอฟท์ วิสิโอ ชื่อ LogicTool ซึ่งโมดูลดังกล่าวจะถูกแสดงในส่วนออกแบบวงจรตรรกะเชิงผสมแบบกราฟิก เพื่อให้ผู้ออกแบบเรียกใช้เมื่อต้องการที่จะสังเคราะห์วงจร เครื่องมือสังเคราะห์วงจรถูกออกแบบให้รับอินพุตเป็นวงจรตรรกะเชิงผสมที่ได้รับการออกแบบในรูปแบบของกราฟิก โดยส่วนออกแบบวงจรตรรกะเชิงผสมแบบกราฟิก และเครื่องมือสังเคราะห์วงจรจะสังเคราะห์วงจรและให้เอาต์พุตอยู่ในรูปแบบไฟล์ข้อความของภาษาเนทลิสต์ตามมาตรฐานของ ISCAS85 และคอลเล็กชันสำหรับเก็บชื่อและค่าความจริงของสายสัญญาณทั้งหมด โดยเครื่องมือสังเคราะห์วงจรในวิทยานิพนธ์นี้ได้ถูกออกแบบให้ประกอบไปด้วยส่วนของโปรแกรมหนึ่งส่วน คือ Synthesis()

และฟังก์ชันหนึ่ง ฟังก์ชัน คือ Synthesizer(ByVal y As Integer) ซึ่งมีลักษณะของการทำงานดังแสดงในรูปที่ 4.41



รูปที่ 4.41 ขั้นตอนการออกแบบเครื่องมือสังเคราะห์วงจร

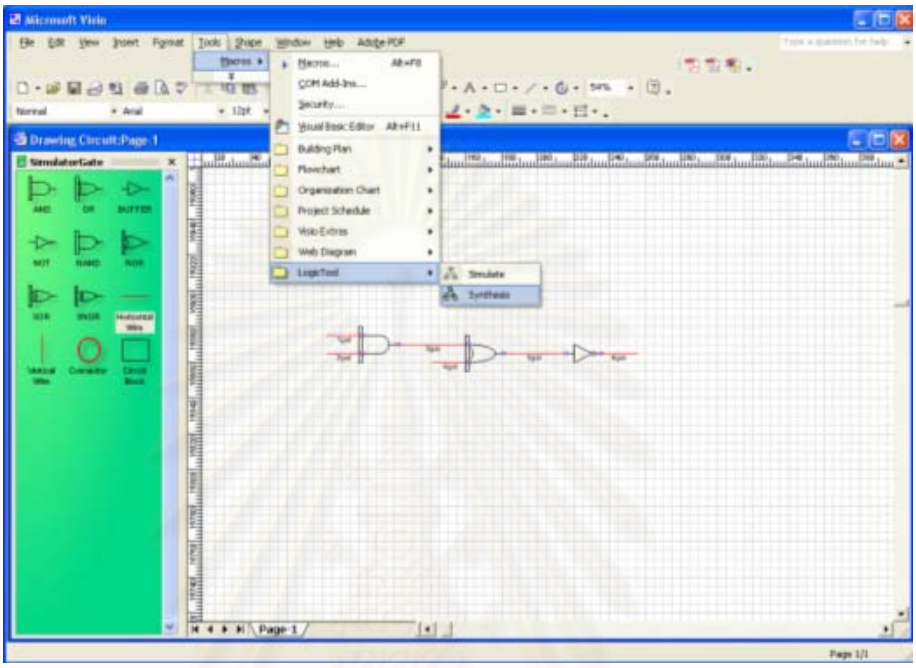
4.2.1โปรแกรม Synthesis()

เป็นส่วนของโปรแกรมที่ทำหน้าที่แสดงหน้าต่างโต้ตอบสำหรับให้ผู้ออกแบบฮาร์ดแวร์กำหนดจุดประสงค์สำหรับการสังเคราะห์วงจรว่าต้องการสังเคราะห์วงจรเพื่อนำไปจำลองการทำงานหรือสังเคราะห์วงจรเพื่อสร้างวงจรบล็อก โดยส่วนของโปรแกรมนี้จะถูกเรียกใช้จากส่วนออกแบบวงจรตรรกะเชิงผสมแบบกราฟิก ดังแสดงในรูปที่ 4.42 เมื่อส่วนของโปรแกรมถูกเรียกใช้ส่วนของโปรแกรมจะแสดงหน้าต่างโต้ตอบเพื่อสอบถามถึงจุดประสงค์การสังเคราะห์วงจรจากผู้ออกแบบฮาร์ดแวร์ ดังแสดงในรูปที่ 4.43 เมื่อผู้ออกแบบฮาร์ดแวร์กำหนดวัตถุประสงค์ของการสังเคราะห์วงจรไม่ว่าจะเป็นแบบการสังเคราะห์วงจรเพื่อนำไปจำลองการทำงานหรือการสังเคราะห์วงจรเพื่อสร้างวงจรบล็อก ส่วนของโปรแกรมนี้จะไปเรียกฟังก์ชัน Synthesizer ทำงานพร้อมทั้งส่งค่าของการเลือกวัตถุประสงค์การสังเคราะห์วงจรไปยังฟังก์ชันดังกล่าว

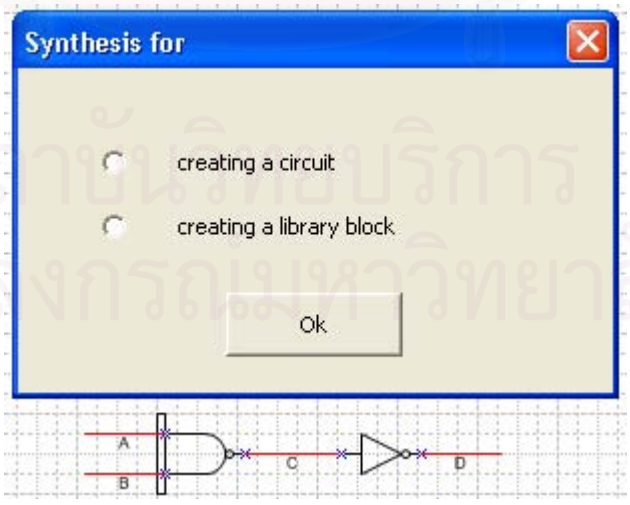
4.2.2 ฟังก์ชัน Synthesizer(ByVal y As Integer)

ฟังก์ชัน Synthesizer จะถูกเรียกใช้งานจากส่วนของโปรแกรม Synthesis โดยส่วนของโปรแกรมนี้จะส่งค่าของการเลือกซึ่งค่าดังกล่าวคือ วัตถุประสงค์ของการสังเคราะห์วงจรโดยถ้าค่าดังกล่าวมีค่าเป็นศูนย์จะแสดงถึงผู้ออกแบบฮาร์ดแวร์ต้องการสังเคราะห์วงจรเพื่อจำลองการทำงาน

แต่ถ้าค่าดังกล่าวเป็นหนึ่งจะแสดงถึงว่าผู้ออกแบบฮาร์ดแวร์ต้องการสังเคราะห์วงจรเพื่อสร้างวงจรบล็อก ซึ่งมีรายละเอียดการทำงานดังต่อไปนี้

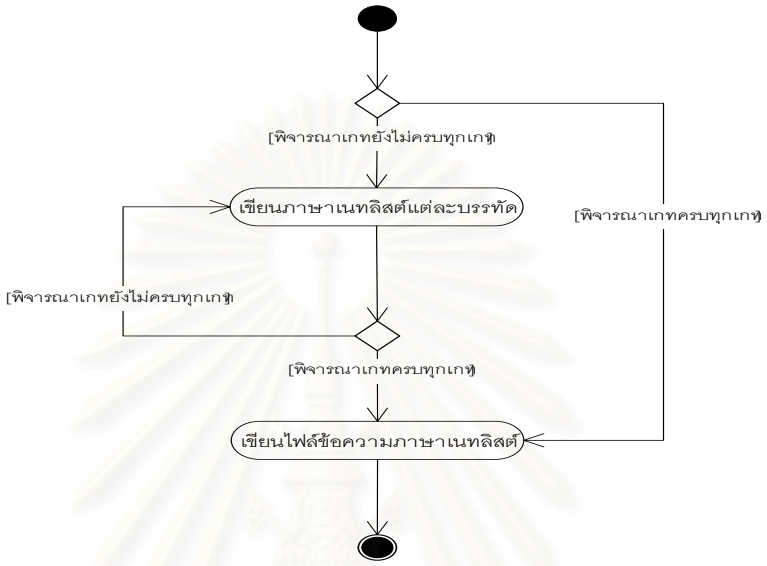


รูปที่ 4.42 การเรียกใช้งานส่วนของโปรแกรม Synthesis จากส่วนออกแบบวงจรตรรกะเชิงผสมแบบกราฟิก



รูปที่ 4.43 หน้าต่างโต้ตอบเพื่อให้ผู้ออกแบบฮาร์ดแวร์กำหนดวัตถุประสงค์ของการสังเคราะห์วงจร

4.2.2.1 การสังเคราะห์วงจรเพื่อจำลองการทำงาน แสดงการทำงานดังรูปที่ 4.44 การทำงานของฟังก์ชันในส่วนนี้จะถูกเรียกใช้ให้ทำงานเมื่อค่าที่ถูกส่งจากส่วนของโปรแกรม Synthesis มีค่าเป็นศูนย์โดยการทำงานจะเริ่มจากการพิจารณารูปร่างที่ถูกออกแบบบนส่วนการออกแบบวงจรตรรกะเชิงผสมแบบกราฟิกเพื่อตรวจสอบ



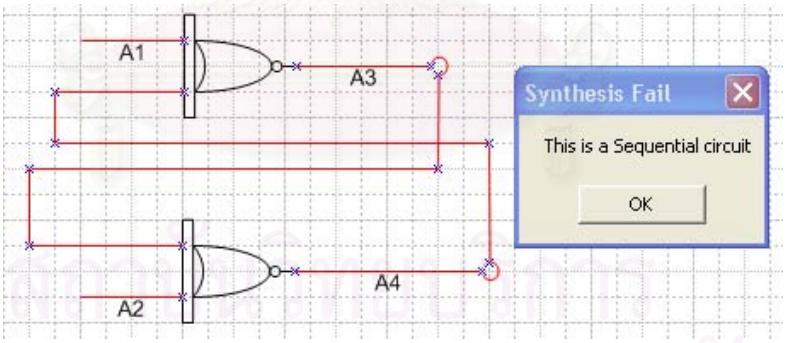
รูปที่ 4.44 ขั้นตอนการทำงานของฟังก์ชัน Synthesizer เพื่อจำลองการทำงาน

ว่าเป็นรูปร่างสองมิติ ซึ่งคือรูปร่างของรูปร่างหลัก แอนด์ ออร์ บัฟเฟอร์ ตัวผกผัน แนนด์ นอร์ ออร์เฉพาะ ออร์ไม่เฉพาะ จุดเชื่อมต่อและวงจรบล็อก แต่ต้องไม่ใช่รูปร่างหลักของ Connector ซึ่งถ้าฟังก์ชันพิจารณาแล้วว่าไม่ใช่รูปร่างสองมิติหรือเป็นรูปร่างของรูปร่างหลักจุดเชื่อมต่อก็จะไม่ทำสิ่งใด แต่ถ้าฟังก์ชันพิจารณาแล้วพบว่าเป็นรูปร่างของรูปร่างหลักดังกล่าว ก็จะเริ่มพิจารณาถึงจุดเชื่อมต่อว่ารูปร่างดังกล่าวมีจุดเชื่อมต่อทางด้านอินพุตหรือไม่และมีจำนวนเท่าใด ถ้าปรากฏว่ามีก็จะตรวจสอบค่าของคุณสมบัติของรูปร่าง คือ Prop.PrimaryInput ซึ่งจะบอกถึงว่ารูปร่างที่เชื่อมต่ออยู่กับรูปร่างที่กำลังพิจารณาเป็นอินพุตหลัก หลังจากนั้นฟังก์ชันก็จะเก็บชื่อและค่าความจริงของรูปร่างที่เชื่อมต่ออยู่เป็นอินพุต โดยที่ถ้าฟังก์ชันพิจารณาแล้วว่าอินพุตดังกล่าวเป็นอินพุตหลักจะถูกกำหนดค่าความจริงให้เป็นเท็จ แต่ถ้าเป็นอินพุตใดๆ จะเก็บค่าความจริงตามที่เป็นอยู่ หลังจากนั้นจะพิจารณาจุดเชื่อมต่อทางด้านเอาต์พุตของรูปร่างสองมิติที่กำลังพิจารณาอยู่ว่ามีหรือไม่และมีจำนวนเท่าใด ซึ่งถ้ามีก็จะตรวจสอบค่าของคุณสมบัติของรูปร่าง คือ Prop.PrimaryOutput ซึ่งจะบอกถึงว่ารูปร่างที่เชื่อมต่ออยู่กับรูปร่างที่กำลังพิจารณาเป็นเอาต์พุตหลักหรือไม่ หลังจากนั้นฟังก์ชันก็จะเก็บชื่อและค่าความจริง

ของรูปร่างที่เชื่อมต่ออยู่เป็นเอาต์พุต เมื่อฟังก์ชันพิจารณาจุดเชื่อมต่อเสร็จแล้ว ฟังก์ชันจะเก็บชื่อของรูปร่างหลักของรูปร่างสองมิติที่กำลังพิจารณาอยู่ ซึ่งเมื่อพิจารณารูปร่างสองมิติที่กำลังถูกพิจารณาเรียบร้อยหนึ่งรูปร่าง ตัวสังเคราะห์วงจรจะเขียนภาษาเนทลิสต์ในรูปแบบมาตรฐาน ISCAS85 ซึ่งมีรูปแบบ คือ

เอาต์พุต = ชื่อเกท(อินพุต1, อินพุต2, ...)

ภาษาเนทลิสต์ของรูปร่างสองมิติที่ถูกพิจารณาที่ถูกสร้างขึ้นจะถูกเก็บอยู่ในคอเลกชันชนิดสายอักขระ หลังจากนั้นจะเวียนพิจารณารูปร่างสองมิติที่ถูกออกแบบบนส่วนการออกแบบวงจรตรรกะเชิงผสมแบบกราฟิกจนครบทุกรูปร่างเพื่อสร้างภาษาเนทลิสต์และเก็บไว้ในคอเลกชันชนิดสายอักขระ หลังจากนั้นฟังก์ชันจะตรวจสอบว่าวงจรที่ผู้ออกแบบฮาร์ดแวร์ออกแบบไว้นั้นเป็นวงจรตรรกะเชิงผสมหรือไม่ ซึ่งจะพิจารณาคอเลกชันที่เก็บภาษาเนทลิสต์ไว้แล้วจะเริ่มพิจารณาอินพุตทุกอินพุตของแต่ละเกทว่าเป็นอินพุตหลักหรือเป็นเอาต์พุตของภาษาเนทลิสต์ที่ถูกพิจารณาแล้วหรือไม่ ถ้าพบว่าอินพุตที่กำลังพิจารณาเป็นอินพุตที่ไม่ใช่อินพุตหลักหรือเป็นเอาต์พุตของภาษาเนทลิสต์ที่ถูกพิจารณาแล้วจะแสดงกล่องข้อความเพื่อเตือนว่าไม่สามารถสังเคราะห์วงจรได้เนื่องจากเป็นวงจรที่ไม่ใช่วงจรตรรกะเชิงผสม ดังแสดงในรูปที่ 4.45



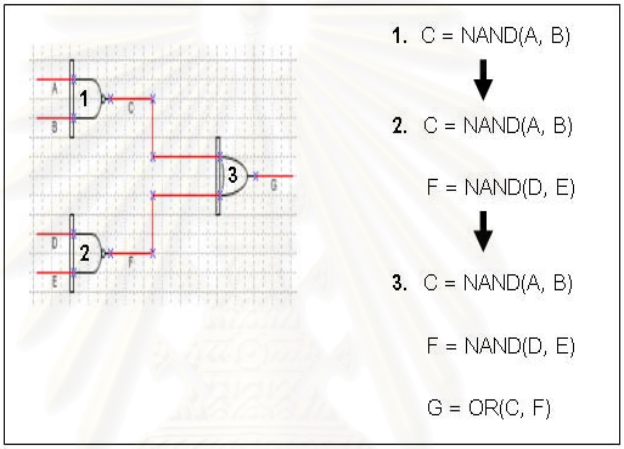
รูปที่ 4.45 กล่องข้อความแจ้งผู้ออกแบบฮาร์ดแวร์เมื่อวงจรนั้นไม่ใช่วงจรตรรกะเชิงผสม

ภาษาเนทลิตส์แต่ละแถวในคอเลกชันที่ถูกพิจารณาไม่พบปัญหาดังกล่าวจะถูกนำไปเขียนลงในตัวแปรชนิดสายอักขระ ซึ่งกระบวนการนี้จะถูกทำซ้ำจนกระทั่งภาษาเนทลิตส์ทุกแถวในคอเลกชันได้รับการพิจารณาจนครบถ้วน หลังจากนั้นฟังก์ชันจะนำตัวแปรสายอักขระมาเขียนเป็นไฟล์ชนิดไฟล์ข้อความ ซึ่งเป็นผลลัพธ์ที่ได้จากเครื่องมือสังเคราะห์วงจร มีลักษณะดังรูปที่ 4.46 ส่วนรูปที่ 4.47 แสดงตัวอย่างขั้นตอนการสังเคราะห์วงจรจากวงจรที่ได้ออกแบบไว้ เป็นไฟล์ข้อ

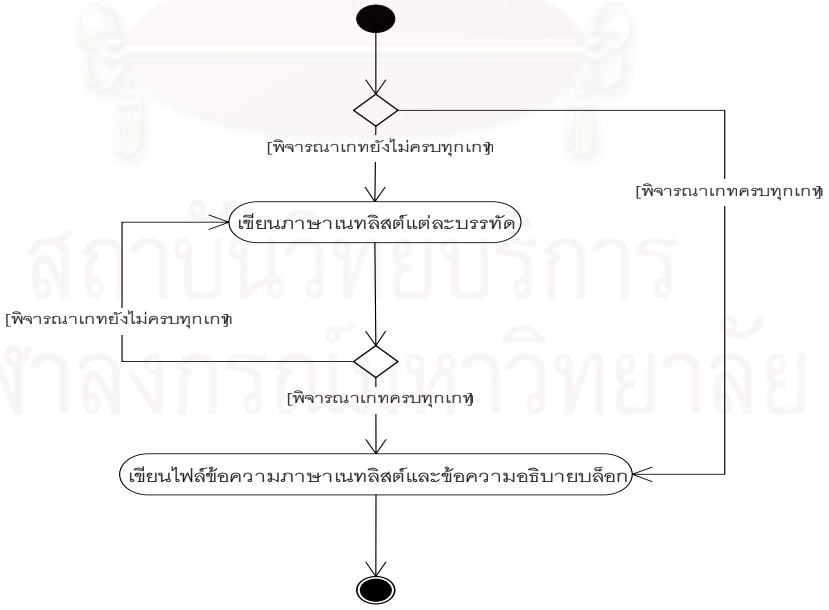
ความภาษาเนทลิสต์ การสังเคราะห์วงจรเพื่อสร้างวงจรบล็อก แสดงการทำงานดังรูปที่ 4.48

```
10gat  =  NAND(1gat, 3gat)
11gat  =  NAND(3gat, 6gat)
16gat  =  NAND(2gat, 11gat)
19gat  =  NAND(11gat, 7gat)
22gat  =  NAND(10gat, 16gat)
23gat  =  NAND(16gat, 19gat)
```

รูปที่ 4.46 ไฟล์ข้อความที่ถูกสร้างขึ้นจากเครื่องมือสังเคราะห์วงจร

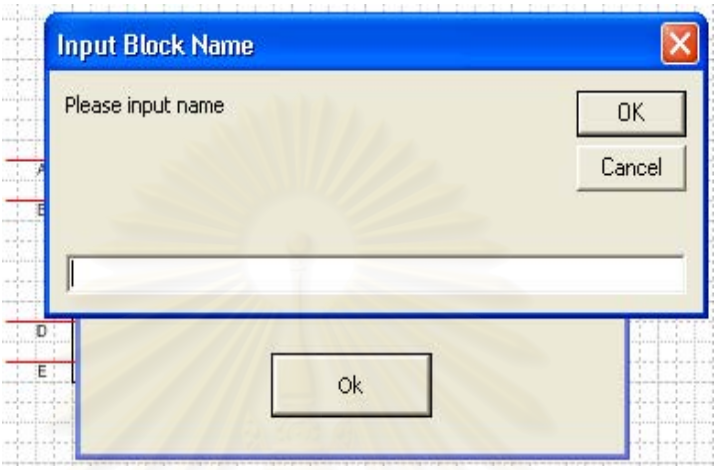


รูปที่ 4.47 ขั้นตอนการสังเคราะห์วงจรจากวงจรที่ได้ออกแบบเป็นไฟล์ข้อความภาษาเนทลิสต์



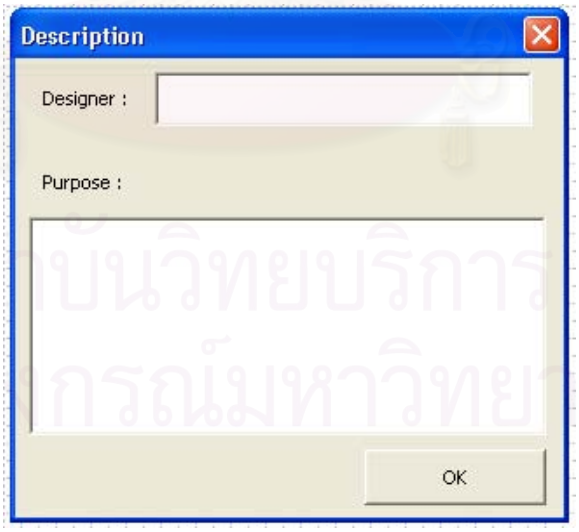
รูปที่ 4.48 ขั้นตอนการทำงานของฟังก์ชัน Synthesizer สำหรับสร้างวงจรบล็อก

การทำงานของฟังก์ชันในส่วนนี้จะถูกเรียกใช้ให้ทำงานเมื่อค่าที่ถูกส่งจากส่วนของโปรแกรม Synthesis มีค่าเป็น 1 โดยการทำงานจะเริ่มจากการแสดงหน้าต่างโต้ตอบเพื่อให้ผู้ออกแบบฮาร์ดแวร์กำหนดชื่อของวงจรบล็อกที่ต้องการจะสังเคราะห์วงจร ดังแสดงในรูปที่ 4.49



รูปที่ 4.49 หน้าต่างโต้ตอบเพื่อกำหนดชื่อของวงจรบล็อกที่ต้องการสังเคราะห์วงจร

เมื่อผู้ออกแบบฮาร์ดแวร์กำหนดชื่อให้กับวงจรบล็อกแล้วจะปรากฏหน้าต่างตอบโต้เพื่อให้ผู้ออกแบบฮาร์ดแวร์กำหนดชื่อของผู้ที่ออกแบบวงจรและวัตถุประสงค์ของการออกแบบวงจร ดังแสดงในรูปที่ 4.50



รูปที่ 4.50 หน้าต่างตอบโต้เพื่อกำหนดชื่อผู้ออกแบบฮาร์ดแวร์และวัตถุประสงค์ของวงจร

การทำงานของฟังก์ชันจะคล้ายกับการสังเคราะห์วงจรเพื่อจำลองการทำงานจะมีความแตกต่างกันเพียงแค่หลังจากที่ได้ผลลัพธ์เป็นไฟล์ข้อความที่เป็นชื่อของวงจร

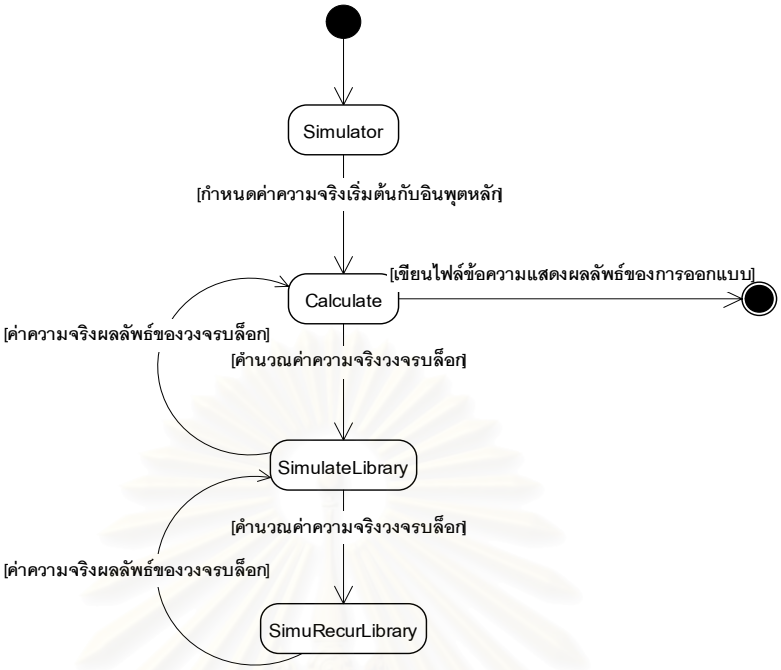
บล็อกเป็นที่เรียบร้อย ฟังก์ชันจะเขียนไฟล์ข้อความอีกไฟล์หนึ่งซึ่งเป็นไฟล์ของภาษาเนทลิสต์ ซึ่งเหมือนกับไฟล์ข้อความที่เป็นผลลัพธ์ของตัวสังเคราะห์วงจรแต่ต่างกันตรงที่จะมีการเขียนข้อความเพิ่มเติมก่อนจะเริ่มต้นภาษาเนทลิสต์ โดยข้อความที่เขียน คือ ชื่อของอินพุตหลักและเอาต์พุตหลักทั้งหมดที่ถูกออกแบบขึ้นในวงจร ไฟล์ข้อความดังกล่าวจะมีชื่อเหมือนกับชื่อที่ผู้ออกแบบฮาร์ดแวร์กำหนดให้กับวงจรบล็อกและจะถูกเพิ่มคำว่า “Detail” แนบกับท้ายของไฟล์

4.3 **เครื่องมือจำลองการทำงาน** (Simulator)

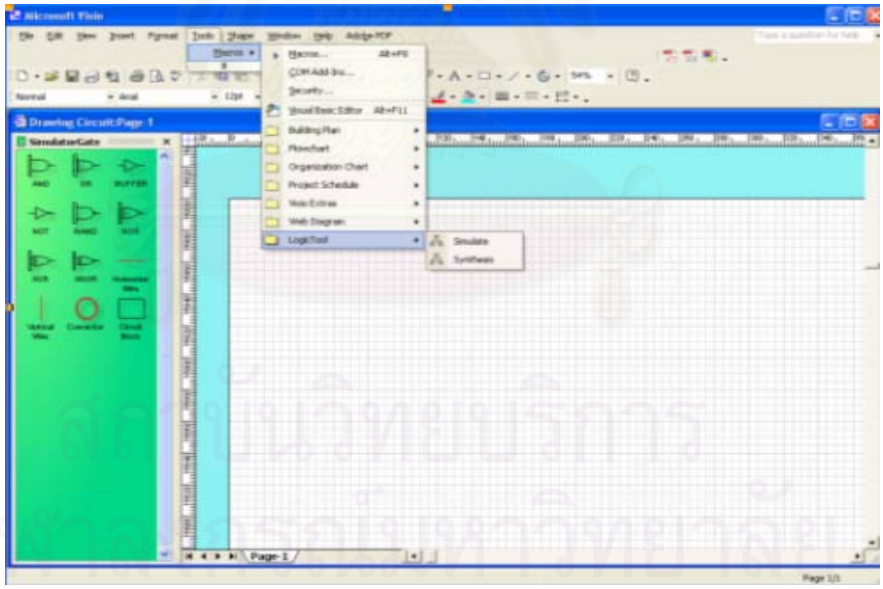
เครื่องมือจำลองการทำงานถูกออกแบบและสร้างขึ้นภายในโมดูล (Module) ของโปรแกรมไมโครซอฟท์ วิสิโอ ชื่อ LogicTool ซึ่งโมดูลดังกล่าวจะถูกแสดงในส่วนออกแบบวงจรตรรกะเชิงผสมแบบกราฟิก เพื่อให้ผู้ออกแบบเรียกใช้เมื่อต้องการที่จะจำลองการทำงาน เครื่องมือจำลองการทำงานถูกออกแบบให้รับอินพุตเป็นไฟล์ชนิดข้อความของวงจรตรรกะเชิงผสมซึ่งเกิดจากตัวสังเคราะห์วงจรพร้อมทั้งคอลเล็กชันที่เก็บชื่อและค่าความจริงของสัญญาณทั้งหมดที่ถูกใช้ในการออกแบบวงจรตรรกะเชิงผสม ซึ่งเมื่อเครื่องมือจำลองการทำงานได้จำลองการทำงานแล้วจะให้ผลลัพธ์ในรูปแบบของไฟล์ชนิดข้อความที่แสดงค่าความจริงของสัญญาณทุกสัญญาณที่ถูกใช้ในการออกแบบวงจรตรรกะเชิงผสมเมื่อผ่านเกทแต่ละเกท โดยตัวจำลองการทำงานที่ถูกออกแบบและสร้างขึ้นในวิทยานิพนธ์นี้ประกอบด้วยส่วนของโปรแกรมจำนวนหนึ่งส่วนของโปรแกรม คือโปรแกรม Simulator() และฟังก์ชันจำนวนสามฟังก์ชัน คือ ฟังก์ชัน Calculate() SimulateLibrary() และ SimuRecurLibrary(recurBlockArray() As String, ByVal x As Integer, ByVal y As Boolean, ByVal z As Integer) ซึ่งมีลักษณะของการทำงานดังแสดงในรูปที่ 4.51

4.3.1 โปรแกรม Simulator()

เป็นส่วนของโปรแกรมที่จะถูกเรียกใช้จากส่วนออกแบบวงจรตรรกะเชิงผสมแบบกราฟิก ดังแสดงในรูปที่ 4.52 หลังจากส่วนของโปรแกรมถูกเรียกใช้แล้ว ส่วนของโปรแกรมจะอ่านไฟล์ชนิดข้อความที่เป็นภาษาเนทลิสต์ของวงจรตรรกะเชิงผสมที่ได้ออกแบบ และสังเคราะห์วงจรไว้ หลังจากนั้น ส่วนของโปรแกรมจะนับจำนวนของอินพุตหลักของวงจรเพื่อนำไปกำหนดค่าความจริงเริ่มต้นให้กับอินพุตหลักทุกตัว โดยหลักการของการกำหนดค่าความจริงให้กับอินพุตหลักสามารถแบ่งได้เป็นสองประเภท คือ



รูปที่ 4.51 ขั้นตอนการออกแบบของเครื่องมือจำลองการทำงาน



รูปที่ 4.52 การเรียกใช้งานส่วนของโปรแกรม Simulator

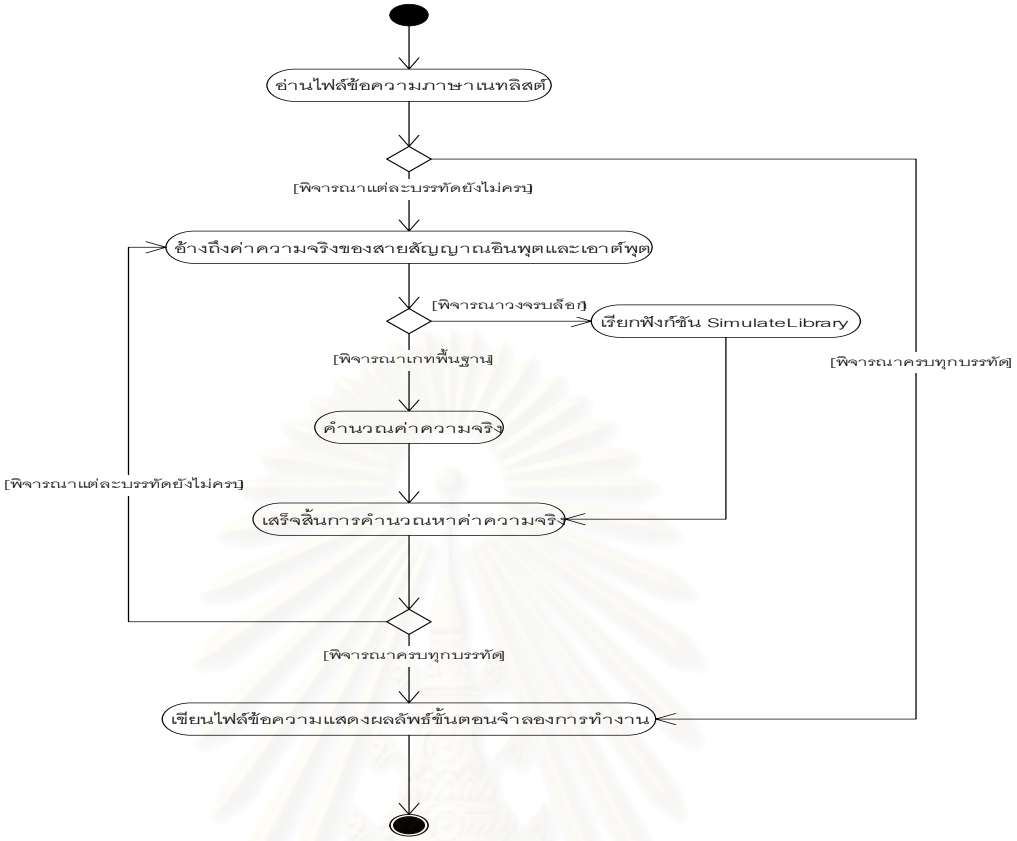
4.3.1.1 ประเภทอินพุตหลักน้อยกว่าหรือเท่ากับเจ็ดบิต เมื่ออินพุตหลักมีจำนวนน้อยกว่าหรือเท่ากับเจ็ดบิตส่วนของโปรแกรมนี้จะกำหนดค่าความจริงให้กับอินพุตหลัก โดยกำหนดค่าความจริงตามหลักของการนับลำดับเลขฐานสอง เมื่อมีการกำหนดค่าความจริงให้กับอินพุตหลักหนึ่งลำดับ ส่วนของโปรแกรมนี้จะเรียกฟังก์ชัน

Calculate() ให้ทำงานจนเสร็จ แล้วก็จะกำหนดค่าความจริงให้กับอินพุตหลักหนึ่งลำดับถัดไป แล้วส่วนของโปรแกรมนี้จะเรียกฟังก์ชัน Calculate() ให้ทำงานจนเสร็จ แล้วก็จะกำหนดค่าความจริงให้กับอินพุตหลักไปเรื่อยๆ จนกระทั่งทุกบิตมีค่าความจริงเป็นจริงหมด ส่วนของโปรแกรมจึงนำผลลัพธ์ที่ได้จากการเรียกฟังก์ชัน Calculate() แต่ละครั้งที่ถูกเก็บไว้ในตัวแปรชนิดอักขระมาเขียนเป็นไฟล์ผลลัพธ์ของการคำนวณชนิดข้อความ

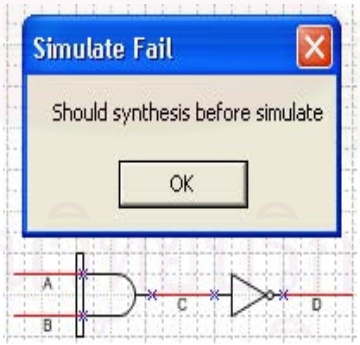
4.3.1.2 ประเภทอินพุตหลักมากกว่าเจ็ดบิต เมื่ออินพุตหลักมีจำนวนมากกว่าเจ็ดบิตส่วนของโปรแกรมนี้จะกำหนดค่าความจริงให้กับอินพุตหลักโดยใช้วิธีการสุ่มค่าความจริงให้กับอินพุตหลักทุกค่าเพียงหนึ่งครั้ง สำหรับวิธีการสุ่มค่าความจริงทำโดยเรียกใช้ฟังก์ชัน Rnd() ของโปรแกรมวิชวลเบสิกสำหรับแอพลิเคชั่น จากนั้นส่วนของโปรแกรมนี้จะเรียกฟังก์ชัน Calculate() ให้ทำงานจนเสร็จ จากนั้นส่วนของโปรแกรมจึงนำผลลัพธ์ที่ได้จากการเรียกฟังก์ชัน Calculate() ที่ถูกเก็บไว้ในตัวแปรชนิดอักขระมาเขียนเป็นไฟล์ผลลัพธ์ของการคำนวณชนิดข้อความ

4.3.2 ฟังก์ชัน Calculate()

เป็นฟังก์ชันที่จะถูกเรียกใช้จากส่วนของโปรแกรม Simulate() เพื่อใช้ในการคำนวณหาค่าความจริงของเกทแต่ละเกทที่ได้ถูกออกแบบไว้ โดยมีลักษณะของการทำงานดังแสดงในรูปที่ 4.53 หลังจากที่อินพุตหลักได้ถูกกำหนดค่าความจริงเริ่มต้นเรียบร้อย ฟังก์ชันนี้จะเริ่มต้นทำงานจากการอ่านไฟล์ชนิดข้อความที่เป็นภาษาเนทลิสต์ของวงจรตรรกะเชิงผสมที่ได้ออกแบบและสังเคราะห์วงจรไว้ โดยที่แต่ละบรรทัดของไฟล์คือการออกแบบของเกทหนึ่งเกทที่ประกอบด้วยเอาต์พุต ชื่อเกท และอินพุตที่เกี่ยวข้องจะถูกนำไปเก็บในคอลเล็กชัน เพื่อจะพิจารณาคำนวณค่าความจริงทีละเกท ซึ่งเมื่อเริ่มต้นพิจารณาค่าความจริงทีละเกทจะพิจารณาจากคอลเล็กชันลำดับแรกสุด โดยจะนำชื่อของอินพุตที่เกี่ยวข้องและเอาต์พุตของบรรทัดที่กำลังพิจารณามาเปรียบเทียบกับชื่อและอ้างถึงค่าความจริงเริ่มต้นที่ถูกกำหนดในแถวลำดับที่ถูกสร้างขึ้นจากขั้นตอนการสังเคราะห์วงจร ซึ่งถ้าผู้ออกแบบฮาร์ดแวร์มิได้สังเคราะห์วงจรก่อน ก็จะไม่สามารถจำลองการทำงานได้ซึ่งจะมีกล่องข้อความเตือน ดังแสดงดังรูป 4.54 แต่ถ้าผู้ออกแบบฮาร์ดแวร์สังเคราะห์วงจรเรียบร้อยแล้ว เมื่ออินพุตและเอาต์พุตของบรรทัดที่กำลังพิจารณาอ้างถึงค่าความจริงที่ถูกกำหนดแล้ว จะเริ่มพิจารณาที่จำนวนว่าเกทดังกล่าวเป็นวงจรบล็อกหรือไม่



รูปที่ 4.53 ขั้นตอนการทำงานของฟังก์ชัน Calculate



รูปที่ 4.54 กล่องข้อความเตือนผู้ออกแบบฮาร์ดแวร์ให้สังเคราะห์วงจรก่อนจำลองการทำงาน

โดยพิจารณาจากชื่อของเกท ซึ่งถ้าเป็นวงจรบล็อก ฟังก์ชันจะเรียกใช้ฟังก์ชันที่มีชื่อว่า SimulateLibrary() แต่ถ้าเกทของบรรทัดที่กำลังพิจารณานั้นมิใช่วงจรบล็อก ฟังก์ชันนี้จะเริ่มพิจารณาถึงจำนวนอินพุตของเกทซึ่งจำนวนอินพุตมากที่สุดที่ตัวจำลองการทำงานรองรับสำหรับเกทแต่ละชนิดคือเก้าอินพุตยกเว้นตัวผกผันและบัฟเฟอร์ที่จะมีจำนวนอินพุตเพียงหนึ่งอินพุต ถ้าจำนวนอินพุตมีเพียงหนึ่งอินพุต ฟังก์ชันนี้จะพิจารณาถึงชื่อของเกทในบรรทัดที่กำลังพิจารณาว่า

เป็นตัวผกผันหรือบัฟเฟอร์แล้วก็จะสร้างแถวลำดับของวัตถุจากคลาสตัวผกผันหรือบัฟเฟอร์ตามที่ได้ออกแบบไว้ในบทที่สามเพื่อคำนวณหาค่าความจริงของเกทดังกล่าว แต่ถ้าจำนวนอินพุตมีจำนวนตั้งแต่สองอินพุตถึงเก้าอินพุต ฟังก์ชันนี้จะพิจารณาถึงชื่อของเกทในบรรทัดที่กำลังพิจารณาว่าเป็น แอนด์ ออร์ แนนด์ นอร์ ออร์เฉพาะ หรือ ออร์ไม่เฉพาะ แล้วก็จะสร้างแถวลำดับของวัตถุจากคลาสตัวผกผันหรือบัฟเฟอร์ตามที่ได้ออกแบบไว้ในบทที่สามเพื่อคำนวณหาค่าความจริงของเกทดังกล่าว หลังจากคำนวณหาค่าความจริงแล้วผลลัพธ์ของค่าความจริงที่ได้คือค่าความจริงของเอาต์พุตซึ่งจะถูกนำไปเก็บไว้ในแถวลำดับซึ่งใช้เก็บชื่อของสัญญาณและค่าความจริงของสัญญาณ อีกทั้งค่าความจริงของสัญญาณทั้งหมดที่ถูกออกแบบในวงจรจะถูกสร้างขึ้นและเก็บไว้ในตัวแปรชนิดสายอักขระ ดังแสดงตัวอย่างในรูปที่ 4.55

```
10gat  =  NAND(1gat, 3gat)
Value : 1gat is : FALSE
Value : 3gat is : FALSE
Value : 10gat is : TRUE
Value : 6gat is : FALSE
Value : 11gat is : FALSE
Value : 2gat is : FALSE
Value : 16gat is : FALSE
Value : 7gat is : FALSE
Value : 19gat is : FALSE
Value : 22gat is : FALSE
Value : 23gat is : FALSE
```

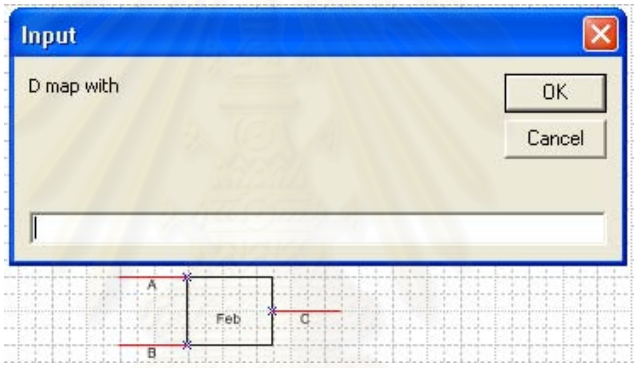
รูปที่ 4.55 ค่าความจริงของสัญญาณที่ถูกสร้างขึ้นและเก็บในตัวแปรสายอักขระ

หลังจากพิจารณาหาค่าความจริงของเกทของวงจรเสร็จหนึ่งบรรทัด ตัวจำลองการทำงานก็จะเรียกวงจรบรรทัดถัดไปที่ถูกเก็บอยู่ในคอลเล็กชันมาพิจารณาหาค่าความจริงต่อไป ซึ่งจะทำเช่นนี้ต่อไปเรื่อยๆ จนกระทั่งหมดบรรทัดสุดท้ายในคอลเล็กชัน หลังจากนั้นจึงนำตัวแปรสายอักขระดังกล่าวไปเขียนเป็นไฟล์ชนิดข้อความซึ่งจะแสดงผลลัพธ์ค่าความจริงของสัญญาณทั้งหมดของวงจรตรรกะเชิงผสมที่ได้ออกแบบไว้

4.3.3 ฟังก์ชัน SimulateLibrary()

เป็นฟังก์ชันที่ถูกเรียกใช้โดยฟังก์ชัน Calculate() เมื่อตำแหน่งของเกทที่ถูกพิจารณาเป็นวงจรบล็อก โดยฟังก์ชันนี้จะเริ่มต้นอ่านไฟล์ชนิดข้อความที่เป็นภาษาเนทลิสต์ของวงจรตรรกะเชิงผสมที่ได้ออกแบบและสังเคราะห์วงจรเป็นวงจรบล็อกไว้ เพื่อเก็บชื่อของอินพุตหลักและเอาต์พุตหลักไว้ หลังจากนั้นฟังก์ชันนี้จะแสดงหน้าต่างโต้ตอบกับผู้ออกแบบฮาร์ดแวร์เพื่อที่จะให้ผู้ออกแบบฮาร์ดแวร์กำหนดชื่อสายสัญญาณให้กับอินพุตหลักและเอาต์พุตหลักของวงจรบล็อกที่ออก

แบบไว้ ดังแสดงในรูปที่ 4.56 นอกจากจะกำหนดชื่อสายสัญญาณให้กับอินพุตหลักและเอาต์พุตหลักวงจรบล็อกแล้ว จะมีการกำหนดค่าความจริงของสายสัญญาณให้กับอินพุตหลักด้วย โดยที่ชื่อและค่าความจริงจะถูกเก็บไว้ในแถวลำดับที่ถูกสร้างขึ้นใหม่ หลังจากนั้นฟังก์ชันนี้จะเริ่มต้นอ่านไฟล์ชนิดข้อความที่เป็นภาษาเนทลิสต์ของวงจรตรรกะเชิงผสมที่ได้ออกแบบและสังเคราะห์วงจรเป็นวงจรบล็อกไว้ โดยที่แต่ละบรรทัดของไฟล์คือการออกแบบของเกทหนึ่งเกทที่ประกอบด้วยเอาต์พุต ชื่อเกท และอินพุตที่เกี่ยวข้องจะถูกนำไปเก็บในคอลเล็กชัน เพื่อจะพิจารณาคำนวณค่าความจริงทีละเกท ซึ่งเมื่อเริ่มต้นพิจารณาค่าความจริงทีละเกทจะพิจารณาจากคอลเล็กชันลำดับแรกสุด โดยจะนำชื่อของอินพุตที่เกี่ยวข้องและเอาต์พุตของบรรทัดที่กำลังพิจารณามาเปรียบเทียบกับชื่อและอ้างถึงค่าความจริง



รูปที่ 4.56 หน้าต่างโต้ตอบเพื่อรับชื่อสายสัญญาณของวงจรบล็อก

เริ่มต้นที่ถูกกำหนดในแถวลำดับที่ถูกสร้างขึ้นใหม่ เมื่ออินพุตและเอาต์พุตของบรรทัดที่กำลังพิจารณาอ้างถึงค่าความจริงที่ถูกกำหนดแล้ว จะเริ่มพิจารณาที่จำนวนว่าเกทดังกล่าวว่าเป็นวงจรบล็อกหรือไม่โดยพิจารณาจากชื่อของเกท ซึ่งถ้าเป็นวงจรบล็อก ซึ่งแสดงถึงว่าการออกแบบวงจรตรรกะเชิงผสมนี้ มีการใช้วงจรบล็อกอีกทีหนึ่ง ฟังก์ชันจะเรียกใช้ฟังก์ชันที่มีชื่อว่า SimuRecurLibrary(recurBlockArray() As String, ByVal x As Integer, ByVal y As Boolean, ByVal z As Integer) แต่ถ้าเกทของบรรทัดที่กำลังพิจารณานั้นมิใช่วงจรบล็อก ฟังก์ชันนี้จะเริ่มพิจารณาถึงจำนวนอินพุตของเกทซึ่งจำนวนอินพุตมากที่สุดที่ตัวจำลองการทำงานรองรับสำหรับเกทแต่ละชนิดคือเก้าอินพุตยกเว้นตัวผกผันและบัฟเฟอร์ที่จะมีจำนวนอินพุตเพียงหนึ่งอินพุต ถ้าจำนวนอินพุตมีเพียงหนึ่งอินพุต ฟังก์ชันนี้จะพิจารณาถึงชื่อของเกทในบรรทัดที่กำลังพิจารณาว่าเป็นตัวผกผันหรือบัฟเฟอร์แล้วก็จะสร้างแถวลำดับของวัตถุจากคลาสตัวผกผันหรือบัฟเฟอร์ตามที่ได้ออกแบบไว้ในบทที่สามเพื่อคำนวณหาค่าความจริงของเกทดังกล่าว แต่ถ้าจำนวนอินพุตมีจำนวนตั้งแต่สองอินพุตถึงเก้าอินพุต ฟังก์ชันนี้จะพิจารณาถึงชื่อของเกทในบรรทัดที่กำลังพิจารณา

ว่าเป็น แอนด์ ออร์ แนนด์ นอร์ ออร์เฉพาะ หรือ ออร์ไม่เฉพาะ แล้วก็จะสร้างแถวลำดับของวัตถุจากคลาสตัวผกผันหรือบัฟเฟอร์ตามที่ได้ออกแบบไว้ในบทที่สาม เพื่อคำนวณหาค่าความจริงของเกทดังกล่าว หลังจากคำนวณหาค่าความจริงแล้วผลลัพธ์ของค่าความจริงที่ได้คือค่าความจริงของเอาต์พุตซึ่งจะถูกนำไปเก็บไว้ในแถวลำดับที่ถูกสร้างขึ้นใหม่ซึ่งใช้เก็บชื่อของสัญญาณและค่าความจริงของสัญญาณ อีกทั้งค่าความจริงของสัญญาณทั้งหมดที่ถูกออกแบบในวงจรจะถูกสร้างขึ้นและเก็บไว้ในตัวแปรชนิดสายอักขระ หลังจากพิจารณาหาค่าความจริงของเกทของวงจรเสร็จหนึ่งบรรทัดตัวจำลองการทำงานก็จะเรียกวงจรบรรทัดถัดไปที่ถูกเก็บอยู่ในคอลเล็กชันมาพิจารณาหาค่าความจริงต่อไป ซึ่งจะทำเช่นนี้ต่อไปเรื่อยๆ จนกระทั่งหมดบรรทัดสุดท้ายในคอลเล็กชัน แล้วจะคืนค่าความจริงของเอาต์พุตหลักกลับไปยังฟังก์ชัน Calculate() เพื่อปรับเปลี่ยนค่าเอาต์พุตที่เกี่ยวข้องในแถวลำดับ

*4.3.3 **ฟังก์ชัน** SimuRecurLibrary(recurBlockArray() As String, ByVal x As Integer, ByVal y As Boolean, ByVal z As Integer)*

เป็นฟังก์ชันที่ถูกเรียกใช้โดยฟังก์ชัน SimulateLibrary() เมื่อตำแหน่งของเกทที่ถูกพิจารณาเป็นวงจรบล็อก โดยฟังก์ชันนี้จะเริ่มต้นอ่านไฟล์ชนิดข้อความที่เป็นภาษาเนทลิสต์ของวงจรตรรกะเชิงผสมที่ได้ออกแบบและสังเคราะห์วงจรเป็นวงจรบล็อกไว้ เพื่อเก็บชื่อของอินพุตหลักและเอาต์พุตหลักไว้ และจะกำหนดชื่อสายสัญญาณอินพุตหลักและเอาต์พุตหลักลักษณะตามลำดับให้กับชื่อสายสัญญาณให้กับอินพุตหลักและเอาต์พุตหลักวงจรบล็อกพร้อมทั้งค่าความจริงของสายสัญญาณด้วย เพื่อสร้างเป็นแถวลำดับใหม่สำหรับใช้ในการคำนวณหาค่าความจริงในฟังก์ชันนี้ หลังจากนั้นฟังก์ชันนี้จะเริ่มต้นอ่านไฟล์ชนิดข้อความที่เป็นภาษาเนทลิสต์ของวงจรตรรกะเชิงผสมที่ได้ออกแบบและสังเคราะห์วงจรเป็นวงจรบล็อกไว้ โดยที่แต่ละบรรทัดของไฟล์คือการออกแบบของเกทหนึ่งเกทที่ประกอบด้วยเอาต์พุต ชื่อเกท และอินพุตที่เกี่ยวข้องจะถูกนำไปเก็บในคอลเล็กชัน เพื่อจะพิจารณาคำนวณค่าความจริงทีละเกท ซึ่งเมื่อเริ่มต้นพิจารณาค่าความจริงทีละเกทจะพิจารณาจากคอลเล็กชันลำดับแรกสุด โดยจะนำชื่อของอินพุตที่เกี่ยวข้องและเอาต์พุตของบรรทัดที่กำลังพิจารณามาเปรียบเทียบกับชื่อและอ้างถึงค่าความจริงเริ่มต้นที่ถูกกำหนดในแถวลำดับที่ถูกสร้างขึ้นใหม่ เมื่ออินพุตและเอาต์พุตของบรรทัดที่กำลังพิจารณาอ้างถึงค่าความจริงที่ถูกกำหนดแล้ว จะเริ่มพิจารณาที่จำนวนว่าเกทดังกล่าวว่าเป็นวงจรบล็อกหรือไม่โดยพิจารณาจากชื่อของเกท ซึ่งถ้าเป็นวงจรบล็อก ซึ่งแสดงถึงว่าการออกแบบวงจรตรรกะเชิงผสมนี้ มีการใช้วงจรบล็อกอีกทีหนึ่ง ฟังก์ชันจะเรียกใช้ฟังก์ชันตัวเองเป็นลักษณะของฟังก์ชันเวียนเกิด แต่ถ้าเกทของบรรทัดที่กำลังพิจารณานั้นมิใช่วงจรบล็อก ฟังก์ชันนี้จะเริ่มพิจารณาถึงจำนวนอินพุตของเกทซึ่งจำนวนอินพุตมากที่สุดที่ตัวจำลองการทำงานรองรับสำหรับเกทแต่ละชนิดคือเก้าอินพุตยกเว้นตัวผกผันและบัฟเฟอร์ที่จะมี

จำนวนอินพุตเพียงหนึ่งอินพุต ถ้าจำนวนอินพุตมีเพียงหนึ่งอินพุต ฟังก์ชันนี้จะพิจารณาถึงชื่อของเกทในบรรทัดที่กำลังพิจารณาว่าเป็นตัวผกผันหรือบัฟเฟอร์แล้วก็จะสร้างแถวลำดับของวัตถุจากคลาสตัวผกผันหรือบัฟเฟอร์ตามที่ได้ออกแบบไว้ในบทที่สาม เพื่อคำนวณหาค่าความจริงของเกทดังกล่าว แต่ถ้าจำนวนอินพุตมีจำนวนตั้งแต่สองอินพุตถึงเก้าอินพุต ฟังก์ชันนี้จะพิจารณาถึงชื่อของเกทในบรรทัดที่กำลังพิจารณาว่าเป็น แอนด์ ออร์ แนนด์ นอร์ ออร์เฉพาะ หรือ ออร์ไม่เฉพาะ แล้วก็จะสร้างแถวลำดับของวัตถุจากคลาสตัวผกผันหรือบัฟเฟอร์ตามที่ได้ออกแบบไว้ในบทที่สาม เพื่อคำนวณหาค่าความจริงของเกทดังกล่าว หลังจากคำนวณหาค่าความจริงแล้วผลลัพธ์ของค่าความจริงที่ได้คือค่าความจริงของเอาต์พุตซึ่งจะถูกนำไปเก็บไว้ในแถวลำดับที่ถูกสร้างขึ้นใหม่ซึ่งใช้เก็บชื่อของสัญญาณและค่าความจริงของสัญญาณ อีกทั้งค่าความจริงของสัญญาณทั้งหมดที่ถูกออกแบบในวงจรจะถูกสร้างขึ้นและเก็บไว้ในตัวแปรชนิดสายอักขระ หลังจากพิจารณาหาค่าความจริงของเกทของวงจรเสร็จหนึ่งบรรทัด ตัวจำลองการทำงานก็จะเรียกวงจรบรรทัดถัดไปที่ถูกจัดเก็บอยู่ภายในคอลเล็กชันมาพิจารณาหาค่าความจริงต่อไป ซึ่งจะทำเช่นนี้ต่อไปเรื่อยๆ จนกระทั่งหมดบรรทัดสุดท้ายในคอลเล็กชัน จากนั้นก็จะทำการคืนค่าความจริงของเอาต์พุตหลักกลับไปยังฟังก์ชัน SimulateLibrary() เพื่อปรับเปลี่ยนค่าเอาต์พุตที่เกี่ยวข้องในแถวลำดับ

บทที่ 5
การทดลองการออกแบบ
วงจรตรรกะเชิงผสมด้วยแนวคิดเชิงวัตถุ

หลังจากการสร้างเครื่องมือออกแบบวงจรตรรกะเชิงผสมแล้ว จะมาถึงขั้นตอนการตรวจสอบความถูกต้องของเครื่องมือดังกล่าว เพื่อเป็นการทบทวนและสร้างความมั่นใจให้กับผู้ออกแบบฮาร์ดแวร์เพื่อนำไปใช้ต่อไป รายละเอียดการทดลองแบบแนวคิดเชิงวัตถุสำหรับออกแบบวงจรตรรกะเชิงผสม โดยได้แบ่งการอธิบายออกเป็นสี่หัวข้อ ได้แก่ วงจรที่ใช้ทดลอง สภาพแวดล้อมการทดลอง ขั้นตอนการทดลอง และการวัดผลการทดลอง

5.1 วงจรที่ใช้ทดลอง

เพื่อตรวจสอบความถูกต้อง จำเป็นที่จะต้องนำวงจรมาตรฐานที่มีอยู่ มาทดลองใช้ในเครื่องมือใหม่นี้ วงจรมาตรฐานที่นำมาใช้ในการทดลอง ได้แก่ วงจร International Symposium Circuit and Systems 1985 : ISCAS85 [24] เนื่องจากวงจรดังกล่าวเป็นวงจรชนิดตรรกะเชิงผสมและอยู่ในรูปแบบภาษาเนทลิสต์โดยที่ภาษาเนทลิสต์เป็นภาษาระดับเกทสำหรับการออกแบบวงจร ซึ่งภาษาเนทลิสต์ของวงจรมีความสอดคล้องกับผลลัพธ์ของเครื่องมือที่สร้างขึ้นในวิทยานิพนธ์นี้ ดังนั้นวิทยานิพนธ์นี้จึงเลือกวงจร ISCAS85 สำหรับใช้ทดลอง ซึ่งวงจร ISCAS85 ประกอบด้วยวงจรจำนวนสิบหนึ่งวงจร คือ C17.BENCH C432.BENCH C499.BENCH C880.BENCH C1355.BENCH C1980.BENCH C2670.BENCH C3540.BENCH C5315.BENCH C6288.BENCH และ C7552.BENCH แต่วงจรที่นำมาใช้ในการทดลองของวิทยานิพนธ์นี้ จะนำมาทดลองหกวงจร เนื่องจากการทดลองนี้ต้องการทดสอบว่าเครื่องมือที่ได้สร้างขึ้นในวิทยานิพนธ์นี้ตามแบบแนวคิดเชิงวัตถุสามารถทำงานได้อย่างถูกต้อง คือ สามารถสร้างภาษาเนทลิสต์ และจำลองการทำงานของวงจรจากวงจรที่ได้ออกแบบไว้อย่างถูกต้องประกอบกับวงจร ISCAS85 ทั้งสิบหนึ่งวงจรซึ่งถูกออกแบบโดยใช้เกทพื้นฐานทั้งแปดชนิดนั้นต่างเป็นวงจรตรรกะเชิงผสมทั้งหมด แต่มีขนาดของวงจร คือ จำนวนเกทที่ใช้ในการออกแบบที่แตกต่างกัน ดังนั้นวิทยานิพนธ์นี้จึงเลือกวงจร ISCAS85 เพียงหกวงจรเพื่อใช้ในการทดสอบ เนื่องจากวงจรทั้งหมดที่เลือกใช้มีขนาดของวงจรที่เหมาะสมกับการทดลอง ซึ่งหกวงจรที่เลือกมาทดสอบ คือ C17.BENCH C432.BENCH C499.BENCH C880.BENCH C1355.BENCH และ C1980.BENCH รูปแบบของวงจรตรรกะเชิงผสม ISCAS85 มีรูปแบบ คือ

เอาต์พุต = ชื่อเกท(อินพุต 1, อินพุต 2, อินพุต 3, ...)

ดังแสดงตัวอย่างในรูปที่ 5.1

```
1   INPUT(1gat)
2   INPUT(2gat)
3   INPUT(3gat)
4   INPUT(6gat)
5   INPUT(7gat)
6   OUTPUT(22gat)
7   OUTPUT(23gat)
8   10gat = nand(1gat, 3gat)
9   11gat = nand(3gat, 6gat)
10  16gat = nand(2gat, 11gat)
11  19gat = nand(11gat, 7gat)
12  22gat = nand(10gat, 16gat)
13  23gat = nand(16gat, 19gat)
14  END
```

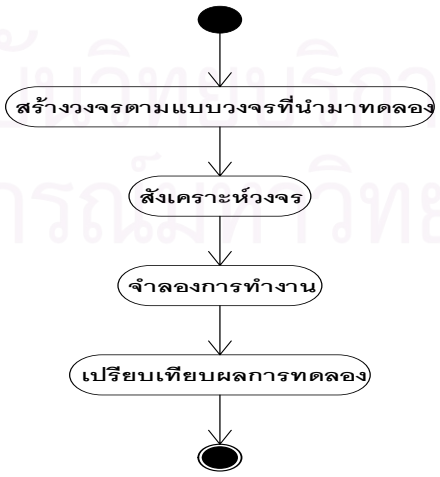
รูปที่ 5.1 ตัวอย่างวงจร C17.BENCH ของวงจร ISCAS85

5.2 สภาพแวดล้อมการทดลอง

การทดลองใช้แบบแนวคิดเชิงวัตถุสำหรับออกแบบวงจรตรรกะเชิงผสม กระทำในเครื่องคอมพิวเตอร์ส่วนบุคคลที่มีหน่วยประมวลผล รุ่น IntelTM Pentium III 1.0 กิกะเฮิรตซ์ (GHz) หน่วยความจำขนาด 512 MB ใช้ระบบปฏิบัติการ รุ่น Microsoft Windows XP Professional และติดตั้งโปรแกรมวิสิโอ รุ่น เอนเทอร์ไพร์ส อาร์คิเท็ค

5.3 ขั้นตอนการทดลอง

ขั้นตอนการทดลองใช้แบบแนวคิดเชิงวัตถุสำหรับออกแบบวงจรตรรกะเชิงผสมดังแสดงในรูปที่ 5.2



รูปที่ 5.2 ขั้นตอนการทดลอง

เริ่มต้นจาก

1) สร้างวงจรตามแบบวงจรที่ต้องการนำมาทดลอง คือ วงจร ISCAS85 จำนวนหกวงจร ด้วยเครื่องมือออกแบบวงจรตรรกะเชิงผสมแบบกราฟิกที่ถูกสร้างขึ้นในวิทยานิพนธ์นี้ เพื่อใช้เป็นอินพุตสำหรับเครื่องมือสังเคราะห์วงจร

2) สังเคราะห์วงจรด้วยเครื่องมือสังเคราะห์วงจรที่ถูกสร้างขึ้นในวิทยานิพนธ์นี้

3) จำลองการทำงานวงจรด้วยเครื่องมือจำลองการทำงานที่ถูกสร้างขึ้นในวิทยานิพนธ์นี้

เมื่อทำการทดลองกับวงจรที่กำหนดไว้จนครบทุกวงจรแล้ว ให้เปลี่ยนไปนำเครื่องมือจำลองการทำงานเครื่องมืออื่นมาจำลองการทำงานวงจรที่นำมาทดลอง แล้วจึงนำผลลัพธ์ที่ได้จากวิธีการทั้งสองชนิดมาเปรียบเทียบกัน

5.4 การเปรียบเทียบผลการทดลอง

การเปรียบเทียบผลการทดลองจะกระทำทั้งหมดสองขั้นตอน คือ

(1) ขั้นสังเคราะห์วงจร

(2) ขั้นจำลองการทำงาน

5.4.1 ขั้นสังเคราะห์วงจร

เมื่อนำวงจรตรรกะเชิงผสม ISCAS85 ที่ใช้ในการทดลองมาออกแบบในส่วนการออกแบบวงจรตรรกะเชิงผสมแบบกราฟิกเป็นที่เรียบร้อยแล้วจึงสังเคราะห์วงจร ผลลัพธ์ของการสังเคราะห์วงจรจากตัวสังเคราะห์วงจรที่ถูกสร้างขึ้นในวิทยานิพนธ์นี้ คือ ไฟล์ชนิดข้อความในรูปแบบของภาษาเนทลิสต์ตามมาตรฐานของ ISCAS85 ดังแสดงตัวอย่างในรูปที่ 5.3

```
10gat = NAND(1gat, 3gat)
11gat = NAND(3gat, 6gat)
16gat = NAND(2gat, 11gat)
19gat = NAND(11gat, 7gat)
22gat = NAND(10gat, 16gat)
23gat = NAND(16gat, 19gat)
```

รูปที่ 5.3 ผลลัพธ์จากการสังเคราะห์วงจร C17.BENCH

ผลลัพธ์จากการสังเคราะห์วงจรจะถูกนำไปเปรียบเทียบความถูกต้องด้วยวิธีการสังเกตและเทียบทีละบรรทัด โดยที่ในแต่ละบรรทัดนั้น อินพุตแต่ละอินพุตสามารถสลับที่กันได้และลำดับบรรทัดสามารถสลับที่กันได้โดยมีกฎว่า การสลับที่ของบรรทัดแต่ละบรรทัดจะไม่ก่อให้เกิดลักษณะวงจรเชิงลำดับ หลังจากเปรียบเทียบความถูกต้องแล้ว ถ้าผลลัพธ์ที่ได้จากการสังเคราะห์วงจรด้วยเครื่องมือสังเคราะห์วงจรที่ถูกสร้างขึ้นในวิทยานิพนธ์นี้เหมือนกับวงจร ISCAS85 ที่นำมาทำการทดลองน่าจะเชื่อได้ว่าเครื่องมือสังเคราะห์วงจรที่ถูกสร้างขึ้นในวิทยานิพนธ์นี้ทำงานถูกต้อง

5.4.2 ขั้นจำลองการทำงาน

ไฟล์ข้อความที่เป็นผลลัพธ์จากการสังเคราะห์วงจรจะถูกนำไปใช้ในการจำลองการทำงานด้วยเครื่องมือจำลองการทำงานที่ถูกสร้างขึ้นในวิทยานิพนธ์นี้ การจำลองการทำงาน คือ การหาค่าความจริงของเกทต่างๆ ในวงจรที่ถูกออกแบบขึ้น เครื่องมือจำลองการทำงานจะให้ผลลัพธ์ในรูปแบบของไฟล์ชนิดข้อความ ที่จะแสดงค่าความจริงของสายสัญญาณทั้งหมดสำหรับเกททั้งหมดที่ถูกออกแบบในวงจร ดังแสดงในรูปที่ 5.4

```
22gat = NAND(10gat, 16gat)
Value : 1gat is : FALSE
Value : 3gat is : FALSE
Value : 10gat is : TRUE
Value : 6gat is : FALSE
Value : 11gat is : TRUE
Value : 2gat is : FALSE
Value : 16gat is : TRUE
Value : 7gat is : FALSE
Value : 19gat is : TRUE
Value : 22gat is : FALSE
Value : 23gat is : FALSE
```

รูปที่ 5.4 ตัวอย่างผลลัพธ์ของการจำลองการทำงานหนึ่งบรรทัดของวงจรทดสอบ

ผลลัพธ์จากขั้นตอนการจำลองการทำงานจะถูกนำไปเปรียบเทียบความถูกต้องกับผลลัพธ์จากการจำลองการทำงานจากวงจรทดสอบเดียวกันโดยเครื่องมือจำลองการทำงานที่ใช้ คือ LogicWorks เวอร์ชัน 4.0 ของบริษัท Capilano Computing Systems Ltd. เนื่องจากเครื่องมือจำลองการทำงานนี้สามารถออกแบบและจำลองการทำงานของวงจรใดๆ ซึ่งรวมถึงวงจรตรรกะเชิงผสมด้วย [21]หน้าต่างการออกแบบแบบกราฟิก และสามารถจำลองการทำงานและแสดงผลลัพธ์ของการจำลองการทำงานได้ ซึ่งองค์ประกอบที่กล่าวมาสามารถใช้เปรียบเทียบผลการทดลองของเครื่องมือที่ได้สร้างขึ้นในวิทยานิพนธ์นี้ได้ ดังนั้นถ้าผลลัพธ์ที่ได้จากการจำลองการทำงานโดยเครื่องมือที่ถูกสร้างขึ้นในวิทยานิพนธ์นี้ ตรงกับผลลัพธ์ที่ได้จากเครื่องมือจำลองการทำงานที่นำมาทดสอบ แสดง

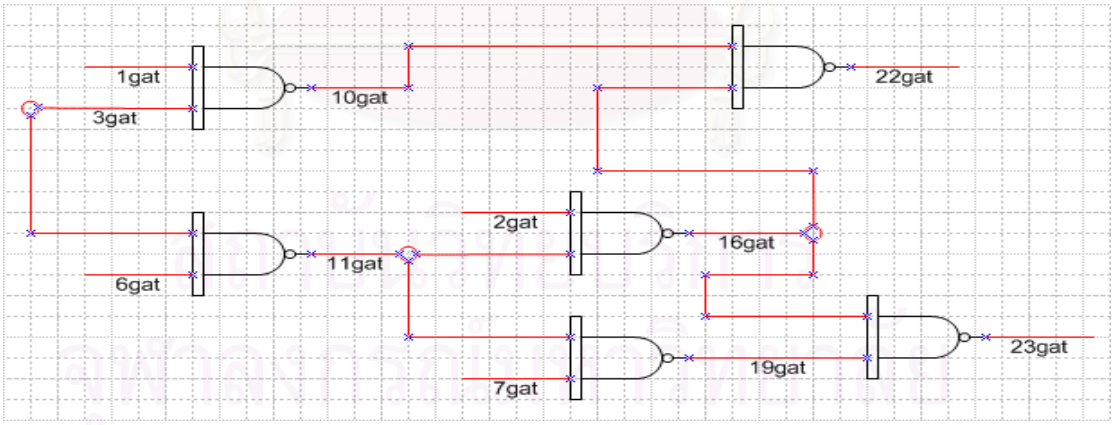
ให้เห็นว่า เครื่องมือจำลองการทำงานสามารถทำงานได้อย่างถูกต้อง และนอกจากที่จะเป็นการแสดงว่าเครื่องมือทั้งหมดที่ถูกสร้างขึ้นในวิทยานิพนธ์นี้สามารถทำงานได้อย่างถูกต้องแล้ว ยังสามารถแสดงได้ว่า การใช้แบบแนวคิดเชิงวัตถุสำหรับการออกแบบวงจรตรรกะเชิงผสมประสบผลสำเร็จอีกด้วย

บทที่ 6
ผลการทดลองการออกแบบวงจรตรรกะเชิงผสม
แบบแนวคิดเชิงวัตถุ

ผลการทดลองการออกแบบวงจรตรรกะเชิงผสมด้วยแนวคิดเชิงวัตถุสำหรับเครื่องมือสำหรับออกแบบวงจรตรรกะเชิงผสมที่พัฒนาขึ้น โดยการทดลองได้นำภาษาเนทลิสต์ของวงจรมาตรฐาน ISCAS85 จำนวนหกวงจร มาทำการทดลอง ตามขั้นตอนที่อธิบายไว้ในบทที่ห้า แต่เนื่องจากผลการทดลองของวงจร C432.BENCH C499.BENCH C880.BENCH C1355.BENCH และ C1980.BENCH ทั้งในขั้นสังเคราะห์วงจรและจำลองการทำงานมีขนาดใหญ่มากกว่าที่จะแสดงในบทนี้ ดังนั้นบทนี้จะนำเสนอผลลัพธ์ของการทดลองในขั้นสังเคราะห์วงจรและจำลองการทำงานของวงจร C17.BENCH และบันทึกผลการทดลองในขั้นสังเคราะห์วงจรและจำลองการทำงานไว้ในแผ่นซีดีรอมแนบไปกับวิทยานิพนธ์นี้ เนื้อหาในบทนี้ได้แบ่งการอธิบายออกเป็นสอง หัวข้อ คือ (1) ผลการสังเคราะห์วงจร และ (2) ผลการทำงาน ซึ่งมีรายละเอียดดังนี้

6.1 ผลการสังเคราะห์วงจร

ผลการสังเคราะห์วงจร คือ ภาษาเนทลิสต์ของวงจรที่ได้นำมาทดลองตามมาตรฐานของ ISCAS85 ซึ่งรูปที่ 6.1 แสดงวงจร C17.BENCH ที่ถูกนำมาออกแบบในส่วนออกแบบวงจรตรรกะเชิงผสมแบบกราฟิก และรูปที่ 6.2 แสดงถึงผลลัพธ์ของขั้นสังเคราะห์วงจรของวงจร C17.BENCH



รูปที่ 6.1 การออกแบบวงจร C17.BENCH ในส่วนออกแบบวงจรตรรกะเชิงผสมแบบกราฟิกที่ได้สร้างขึ้นในวิทยานิพนธ์

```
10gat  =  NAND(1gat, 3gat)
11gat  =  NAND(3gat, 6gat)
16gat  =  NAND(2gat, 11gat)
19gat  =  NAND(11gat, 7gat)
22gat  =  NAND(10gat, 16gat)
23gat  =  NAND(16gat, 19gat)
```

รูปที่ 6.2 ผลลัพธ์ของขั้นสังเคราะห์วงจรของวงจร C17.BENCH

6.2 ผลการทำงาน

ผลการทำงาน คือ ไฟล์ชนิดข้อความที่แสดงผลลัพธ์ค่าความจริงของเกทแต่ละบรรทัดตามลำดับจากภาษาเนทลิสต์ของวงจรซึ่งเป็นผลลัพธ์ที่ได้มาจากเครื่องมือสังเคราะห์วงจร รูปที่ 6.3 เป็นผลลัพธ์จากเครื่องมือจำลองการทำงาน ซึ่งได้จำลองการทำงานจากผลลัพธ์ของขั้นตอนสังเคราะห์วงจรของวงจร C17.BENCH และตารางที่ 6.1 แสดงจำนวนเกทของวงจร เวลาสำหรับการสังเคราะห์วงจร และเวลาสำหรับการจำลองการทำงานวงจรทั้งหกวงจรที่นำมาทดลอง

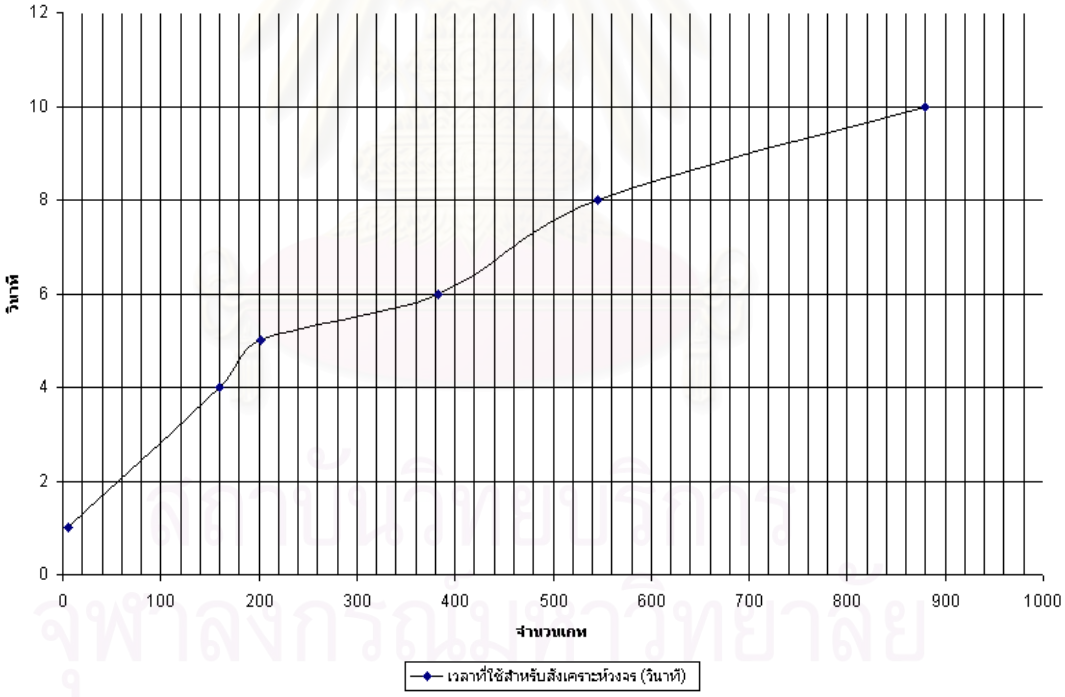
```
10gat  =  NAND(1gat, 3gat)
Value : 1gat is : FALSE
Value : 3gat is : FALSE
Value : 10gat is : TRUE
Value : 6gat is : FALSE
Value : 11gat is : FALSE
Value : 2gat is : FALSE
Value : 16gat is : FALSE
Value : 7gat is : FALSE
Value : 19gat is : FALSE
Value : 22gat is : FALSE
Value : 23gat is : FALSE

11gat  =  NAND(3gat, 6gat)
Value : 1gat is : FALSE
Value : 3gat is : FALSE
Value : 10gat is : TRUE
Value : 6gat is : FALSE
Value : 11gat is : TRUE
Value : 2gat is : FALSE
Value : 16gat is : FALSE
Value : 7gat is : FALSE
Value : 19gat is : FALSE
Value : 22gat is : FALSE
Value : 23gat is : FALSE

16gat  =  NAND(2gat, 11gat)
Value : 1gat is : FALSE
Value : 3gat is : FALSE
Value : 10gat is : TRUE
Value : 6gat is : FALSE
Value : 11gat is : TRUE
Value : 2gat is : FALSE
Value : 16gat is : TRUE
Value : 7gat is : FALSE
Value : 19gat is : FALSE
Value : 22gat is : FALSE
Value : 23gat is : FALSE

19gat  =  NAND(11gat, 7gat)
Value : 1gat is : FALSE
Value : 3gat is : FALSE
Value : 10gat is : TRUE
Value : 6gat is : FALSE
Value : 11gat is : TRUE
Value : 2gat is : FALSE
Value : 16gat is : TRUE
Value : 7gat is : FALSE
Value : 19gat is : TRUE
Value : 22gat is : FALSE
Value : 23gat is : FALSE

22gat  =  NAND(10gat, 16gat)
Value : 1gat is : FALSE
Value : 3gat is : FALSE
Value : 10gat is : TRUE
Value : 6gat is : FALSE
Value : 11gat is : TRUE
Value : 2gat is : FALSE
Value : 16gat is : TRUE
Value : 7gat is : FALSE
Value : 19gat is : TRUE
Value : 22gat is : FALSE
Value : 23gat is : FALSE

23gat  =  NAND(16gat, 19gat)
Value : 1gat is : FALSE
Value : 3gat is : FALSE
Value : 10gat is : TRUE
Value : 6gat is : FALSE
Value : 11gat is : TRUE
Value : 2gat is : FALSE
Value : 16gat is : TRUE
Value : 7gat is : FALSE
Value : 19gat is : TRUE
Value : 22gat is : FALSE
Value : 23gat is : FALSE
```

รูปที่ 6.3 ผลลัพธ์จากการจำลองการทำงานวงจร C17.BENCH

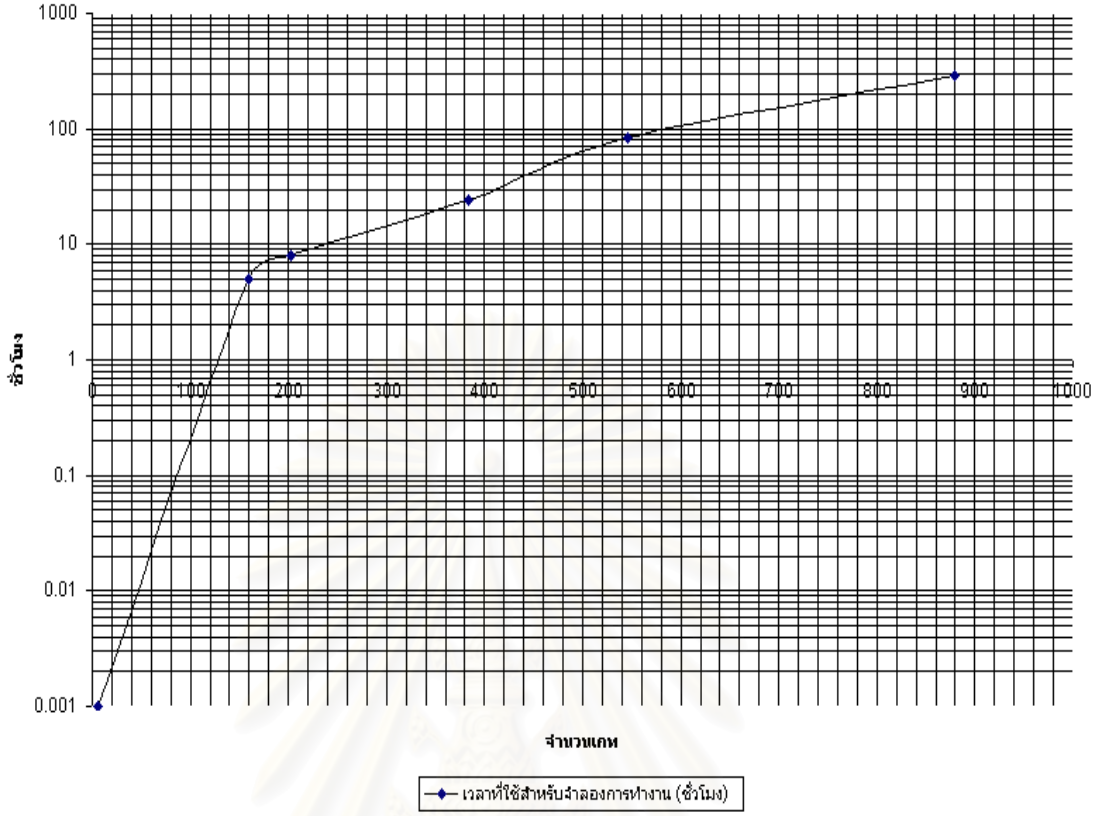
ตารางที่ 6.1 ชื่อวงจร จำนวนเกท เวลาที่ใช้ในการสังเคราะห์และจำลองการทำงาน

ชื่อวงจร	จำนวนเกท	เวลาที่ใช้สำหรับการสังเคราะห์วงจร (วินาที)	เวลาที่ใช้สำหรับจำลองการทำงาน (ชั่วโมง)
C17.BENCH	6	1	0.001
C432.BENCH	160	4	5
C499.BENCH	202	5	8
C880.BENCH	383	6	24
C1355.BENCH	546	8	85
C1908.BENCH	880	10	288

เมื่อนำจำนวนเกทและเวลาที่ใช้สำหรับการสังเคราะห์วงจรมาสร้างเป็นกราฟเชิงเส้นจะมีลักษณะดังแสดงในรูปที่ 6.4 และรูปที่ 6.5 แสดงกราฟเชิงเส้นระหว่างจำนวนเกทและเวลาที่ใช้สำหรับจำลองการทำงาน



รูปที่ 6.4 กราฟเชิงเส้นระหว่างจำนวนเกทและเวลาที่ใช้สำหรับสังเคราะห์วงจร



รูปที่ 6.5 กราฟเชิงเส้นระหว่างจำนวนเกทและเวลาที่ใช้สำหรับจำลองการทำงาน

บทที่ 7
ผลการทดลองการออกแบบวงจรตรรกะเชิงผสมด้วยเครื่องมือชนิดอื่น

การนำเสนอผลการทดลองการออกแบบวงจรมาตรฐาน ISCAS85 จำนวนหกวงจร ในขั้นตอนจำลองการทำงานด้วยเครื่องมือจำลองการทำงานอื่น ตามขั้นตอนที่ได้อธิบายไว้ในบทที่ห้า แต่เนื่องจากผลการทดลองของวงจร C432.BENCH C499.BENCH C880.BENCH C1355.BENCH และ C1980.BENCH ในขั้นจำลองการทำงานมีขนาดใหญ่มากกว่าที่จะแสดงในบทนี้ ดังนั้นบทนี้จะนำเสนอผลลัพธ์ของการทดลองในขั้นจำลองการทำงานของวงจร C17.BENCH และบันทึกผลการทดลองในขั้นจำลองการทำงานไว้ในแผ่นซีดีรอมแนบไปกับวิทยานิพนธ์นี้ เนื้อหาในบทนี้จะอธิบายผลการทำงาน ซึ่งมีรายละเอียดดังนี้

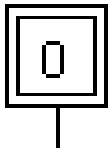
7.1 ผลการทำงาน

ผลการทำงานของวงจรจะถูกสร้างขึ้นจากเครื่องมือออกแบบและจำลองการทำงานชื่อ LogicWorks เวอร์ชัน 4.0 ของบริษัท Capilano Computing Systems Ltd. โดยวงจรมาตรฐาน ISCAS85 ที่นำมาทดลอง จะถูกออกแบบลงบนหน้าต่างออกแบบของโปรแกรม และจำลองการทำงาน โดยอุปกรณ์ที่เกี่ยวข้องในเครื่องมือนี้ที่ใช้ในการทดลอง ประกอบด้วย

- Binary Switch คือ อุปกรณ์สำหรับกำหนดค่าความจริงเริ่มต้นให้กับอินพุตหลักระหว่างค่าความจริง 0 และ 1 โดยผู้ออกแบบฮาร์ดแวร์สามารถกำหนดค่าได้ ดังแสดงสัญลักษณ์ในรูปที่ 7.1

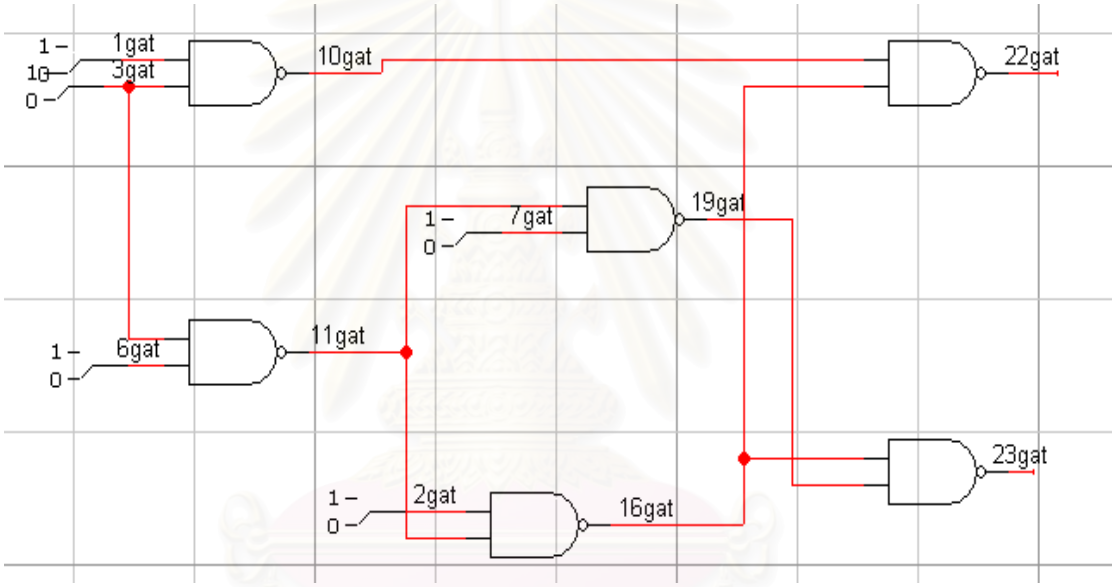
รูปที่ 7.1 สัญลักษณ์ Binary Switch

- Binary Probe คือ อุปกรณ์สำหรับแสดงค่าความจริงของสายสัญญาณใดๆ ที่นำเอา Binary Probe ไปเชื่อมต่อ โดยจะแสดงค่าความจริงในรูปแบบของเลข 1 หรือ เลข 0 ดังแสดงสัญลักษณ์ในรูปที่ 7.2



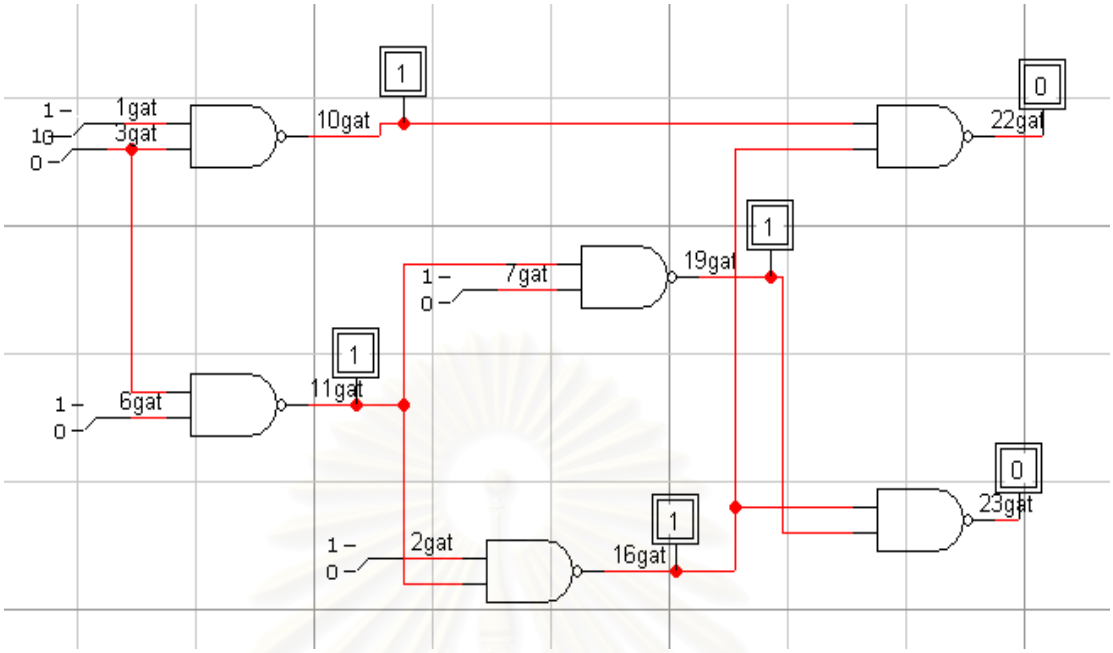
รูปที่ 7.2 สัญลักษณ์ Binary Probe

เมื่อนำวงจร C17.BENCH มาออกแบบในหน้าต่างการออกแบบของโปรแกรม LogicWorks พร้อมทั้งเชื่อมต่อ Binary Switch เข้ากับอินพุตหลักทุกอินพุตมีลักษณะดังรูปที่ 7.3



รูปที่ 7.3 การออกแบบวงจร C17.BENCH บนหน้าต่างออกแบบของวงจร LogicWorks

Binary Probe จำนวนหกตัวจะเชื่อมกับสายสัญญาณเอาต์พุตทุกเส้นของเกททุกเกทในวงจร C17.BENCH ที่ได้ออกแบบไว้ในหน้าต่างของการออกแบบวงจร เพื่อแสดงค่าความจริงของเอาต์พุตของเกทแต่ละเกท ซึ่งค่าความจริงของเอาต์พุตของเกทแต่ละเกท คือ ผลลัพธ์ขั้นตอนจำลองการทำงานด้วยเครื่องมือ LogicWorks ดังแสดงในรูปที่ 7.4



รูปที่ 7.4 ผลลัพธ์จากการจำลองการทำงานของวงจร C17.BENCH

บทที่ 8
การวิเคราะห์ผลการทดลอง

การวิเคราะห์ผลการทดลองวิธีการออกแบบวงจรตรรกะเชิงผสม สามารถแบ่งการอธิบายออกเป็นสองหัวข้อ ได้แก่ บทวิเคราะห์การออกแบบวงจรตรรกะเชิงผสมด้วยเครื่องมือแบบแนวคิดเชิงวัตถุ และการเปรียบเทียบผลการทดลอง

8.1 บทวิเคราะห์การออกแบบวงจรตรรกะเชิงผสมด้วยเครื่องมือแบบแนวคิดเชิงวัตถุ

เครื่องมือออกแบบวงจรตรรกะเชิงผสมแบบแนวคิดเชิงวัตถุที่ได้สร้างขึ้นในวิทยานิพนธ์นี้ ได้นำแนวคิดเชิงวัตถุสามแนวคิดมาใช้ คือ 1) แนวคิดการห่อหุ้ม 2) แนวคิดการถ่ายทอด และ 3) แนวคิดการนำกลับมาใช้ เมื่อนำเครื่องมือสำหรับออกแบบวงจรตรรกะเชิงผสมด้วยแบบแนวคิดเชิงวัตถุที่สร้างขึ้นไปทำการทดลองกับวงจรมาตรฐาน ISCAS85 จำนวนหกวงจรแล้ว บทวิเคราะห์ของเครื่องมือออกแบบแบบแนวคิดเชิงวัตถุโดยอธิบายแต่ละแนวคิดมีรายละเอียดดังนี้

1) แนวคิดการห่อหุ้ม

ส่วนของแนวคิดการห่อหุ้ม คือ คลาสของเกทชนิดต่างๆ ที่ได้นำมาใช้ในการออกแบบวงจร สามารถทำงานได้อย่างถูกต้อง คือ เกทต่างๆ ที่นำมาใช้ในการออกแบบวงจรมาตรฐานจำนวนหกวงจรจะอยู่ในลักษณะวัตถุของคลาสของเกทชนิดที่ได้นำมาใช้สำหรับออกแบบ ซึ่งประกอบด้วยข้อมูลที่จัดเก็บอย่างถูกต้อง และมีการทำงานของพฤติกรรมสำหรับหาค่าความจริงของเกทต่างๆ ที่ใช้ในการออกแบบโดยเกี่ยวข้องกับข้อมูลของวัตถุที่นำมาใช้ได้อย่างถูกต้อง

2) แนวคิดการถ่ายทอด

ส่วนของแนวคิดการถ่ายทอดที่ได้นำมาใช้ในการออกแบบวงจร คือ คลาสของเกทพื้นฐานจำนวนแปดชนิดซึ่งได้รับการถ่ายทอดมาจากคลาส Gate ซึ่งเป็นคลาสเริ่มต้น และแนวคิดการถ่ายทอดยังช่วยให้เกทพื้นฐานทั้งหกชนิด ซึ่งมิได้รวมถึง ตัวผกผันและบัฟเฟอร์ที่รับอินพุตจำนวนเพียงบิตเดียว ที่ใช้ในการออกแบบวงจรตรรกะเชิงผสมซึ่งได้ออกแบบและสร้างขึ้นในเครื่องมือนี้สามารถรองรับอินพุตได้เป็นจำนวนเก้าอินพุตโดยไม่จำเป็นที่เกทพื้นฐานแต่ละชนิดจะต้องมีรูปร่างที่เฉพาะเจาะจงตามจำนวนอินพุตที่ต้องการใช้ในการออกแบบของเกททั้งหกชนิด ประโยชน์ของการนำแนวคิดดังกล่าวมาใช้ คือ สามารถลดจำนวนตัวเลือกของเกทสำหรับใช้ในการออกแบบและลดขนาดของเครื่องมือได้ เกทแต่ละชนิดสามารถรับจำนวนอินพุตได้เป็นจำนวนมากที่สุดเก้าอินพุตโดยที่สัญลักษณ์ของเกทแต่ละชนิดมีเพียงสัญลักษณ์

เดียวเท่านั้น ซึ่งผลที่ได้จากการสังเคราะห์วงจรของเกทและจำนวนอินพุตของเกทแต่ละเกทที่ใช้ในการออกแบบวงจรทั้งหกวงจรให้ผลที่ถูกต้อง

3) แนวคิดการนำกลับมาใช้

วงจรบล็อกถูกออกแบบขึ้น เพื่อใช้สำหรับนำวงจรที่ได้ออกแบบไว้ กลับมาใช้ในการออกแบบวงจรที่มีวงจรดังกล่าวเป็นส่วนประกอบ โดยที่ผู้ออกแบบฮาร์ดแวร์ไม่ต้องออกแบบวงจรในส่วนนั้นๆ อีกครั้งหนึ่ง ซึ่งช่วยลดเวลาของขั้นตอนการออกแบบได้ วงจรบล็อกที่ได้ออกแบบ เมื่อนำไปใช้ในการออกแบบให้ผลการทำงานที่ถูกต้อง คือ วงจรที่ได้ออกแบบไว้เมื่อได้นำไปสังเคราะห์และสร้างขึ้นเป็นวงจรบล็อก และวงจรบล็อกดังกล่าวสามารถคำนวณหาค่าความจริงของเอาต์พุตได้อย่างถูกต้อง

8.2 การเปรียบเทียบผลการทดลอง

เมื่อนำผลการทดลองการออกแบบวงจรมาตรฐานจำนวนหกวงจรจากเครื่องมือออกแบบวงจรตรรกะเชิงผสมแบบแนวคิดเชิงวัตถุที่ได้สร้างขึ้นในวิทยานิพนธ์นี้ เปรียบเทียบกับผลการทดลองจากเครื่องมือออกแบบวงจรตรรกะเชิงผสมเครื่องมืออื่น พบว่าเครื่องมือทั้งสองชนิดให้ผลลัพธ์ของการคำนวณค่าความจริงของเอาต์พุตของวงจรมาตรฐานทั้งหกวงจรที่เหมือนกัน แต่การทำงานของเครื่องมือมีสิ่งที่แตกต่างกัน ซึ่งสามารถสรุปสิ่งที่แตกต่างกันของการทำงานของเครื่องมือ ดังนี้

1) ส่วนออกแบบวงจรแบบกราฟิก เครื่องมือออกแบบแบบแนวคิดเชิงวัตถุที่ได้พัฒนาขึ้นนั้นมีเกทสำหรับใช้ในการออกแบบจำนวนแปดเกท โดยที่เกทแต่ละชนิดจะมีเพียงรูปแบบเดียวซึ่งเกทหกชนิด ยกเว้น ตัวผกผันและบัฟเฟอร์ สามารถรับอินพุตได้จำนวนมากที่สุดเป็นจำนวนเก้าอินพุต ในขณะที่เครื่องมือทั่วไปนั้นแต่ละชนิดของเกทจะมีรูปแบบหลายรูปแบบที่แตกต่างกันตามจำนวนของอินพุต ทำให้การเลือกใช้เกทแต่ละชนิดนั้นนอกจากที่จะเลือกชนิดของเกทแล้ว ยังต้องเลือกรูปแบบของเกทตามจำนวนอินพุตที่ต้องการใช้อีกด้วย ซึ่งเครื่องมือออกแบบแบบแนวคิดเชิงวัตถุช่วยให้การเลือกเกทสำหรับออกแบบวงจรตรรกะเชิงผสมมีความสะดวกยิ่งขึ้น

2) เครื่องมือจำลองการทำงาน เครื่องมือจำลองการทำงานที่สร้างขึ้นภายในเครื่องมือออกแบบแบบแนวคิดเชิงวัตถุนั้นสามารถคำนวณหาค่าความจริงเอาต์พุตของวงจรที่ได้ออกแบบไว้ถูกต้อง แต่เวลาที่ใช้ในการคำนวณหาค่าความจริงเอาต์พุตจะใช้เวลามากขึ้นเมื่อจำนวนของเกทที่ใช้ในการออกแบบวงจรมีเพิ่มขึ้น เมื่อเทียบกับเครื่องมือจำลองการทำงานที่นำมาเปรียบเทียบ

บทที่ 9
บทสรุป

วิทยานิพนธ์นี้ได้ออกแบบและสร้างเครื่องมือสำหรับออกแบบวงจรตรรกะเชิงผสม ซึ่งนำแบบแนวคิดเชิงวัตถุมาใช้ในการออกแบบเครื่องมือ รวมถึงการทดลองเพื่อศึกษาผลของการนำแบบแนวคิดเชิงวัตถุมาออกแบบวงจรตรรกะเชิงผสม เปรียบเทียบกับการออกแบบวงจรตรรกะเชิงผสมด้วยเครื่องมือทั่วไป ที่ได้รับการยอมรับในปัจจุบัน

แนวคิดเชิงวัตถุที่ได้นำมาใช้ออกแบบและสร้างเครื่องมือออกแบบวงจรตรรกะเชิงผสม ประกอบด้วยแนวคิดสามแนวคิด คือ 1) แนวคิดการห่อหุ้ม ซึ่งแนวคิดนี้จะใช้สำหรับออกแบบคลาสของเกทพื้นฐานจำนวนแปดชนิด คือ แอนด์ ออร์ บัฟเฟอร์ ตัวผกผัน แนนด์ นอร์ ออร์เฉพาะ และ ออร์ไม่เฉพาะ ที่ใช้ในการออกแบบวงจรตรรกะเชิงผสม และคลาส Gate ซึ่งเป็นคลาสต้นแบบสำหรับเกทพื้นฐานทั้งแปดชนิด 2) แนวคิดการถ่ายทอด ใช้ในการถ่ายทอดคลาสของเกทพื้นฐานจำนวนแปดชนิดจากคลาส Gate ซึ่งเป็นคลาสต้นแบบ เพื่อใช้ในการคำนวณค่าความจริงเอาต์พุตของเกทชนิดที่ผู้ออกแบบทำการออกแบบไว้ และแนวคิดเชิงวัตถุสุดท้ายที่ได้นำมาใช้ คือ 3) แนวคิดการนำกลับมาใช้ เป็นแนวคิดที่ใช้ในการนำวงจรตรรกะเชิงผสมที่ได้ออกแบบไว้กลับมาใช้อีกครั้งหนึ่ง โดยในเครื่องมือที่สร้างขึ้นนี้แนวคิดดังกล่าวได้แสดงออกในรูปแบบของวงจรบล็อก ซึ่งผู้ออกแบบฮาร์ดแวร์สามารถนำวงจรบล็อกมาใช้ในการออกแบบวงจรใหม่ได้

เครื่องมือออกแบบวงจรตรรกะเชิงผสมแบบแนวคิดเชิงวัตถุที่ได้ออกแบบและสร้างขึ้น ประกอบด้วยเครื่องมือสามเครื่องมือ คือ 1) ส่วนออกแบบวงจรตรรกะเชิงผสมแบบกราฟิก เครื่องมือที่สร้างขึ้นสนับสนุนการออกแบบวงจรตรรกะเชิงผสมแบบกราฟิกด้วยหลักการลากแล้วปล่อย ซึ่งช่วยเพิ่มความสะดวกให้กับการออกแบบวงจร 2) เครื่องมือสังเคราะห์วงจร เมื่อวงจรตรรกะเชิงผสมได้ออกแบบในส่วนออกแบบวงจรแบบกราฟิกแล้ว เครื่องมือสังเคราะห์วงจรจะสังเคราะห์วงจรเพื่อตรวจสอบความถูกต้องของการออกแบบวงจรตรรกะเชิงผสม และสร้างไฟล์ชนิดข้อความ ซึ่งไฟล์ดังกล่าวจะเก็บผลของการออกแบบวงจร ในรูปแบบของภาษาเนทลิสต์ ซึ่งเป็นภาษาระดับเกทตามมาตรฐานของ ISCAS85 และเครื่องมือสุดท้ายที่ได้พัฒนาขึ้นคือ 3) เครื่องมือจำลองการทำงาน เป็นเครื่องมือที่ใช้สำหรับจำลองการทำงานของวงจรเพื่อหาค่าความจริงเอาต์พุตของเกทชนิดต่างๆ ที่ผู้ออกแบบฮาร์ดแวร์ได้ออกแบบวงจรไว้ โดยเครื่องมือนี้จะทำงานหลังจากที่วงจรที่ได้ออกแบบไว้ ผ่านการสังเคราะห์วงจรแล้ว และการจำลองการทำงานของวงจรจะไม่คำนึงถึงดีเลย์

เครื่องมือออกแบบวงจรตรรกะเชิงผสมแบบแนวคิดเชิงวัตถุที่ได้ออกแบบและสร้างขึ้น ได้นำมาทดลองออกแบบวงจรมาตรฐาน ISCAS85 จำนวนหกวงจร คือ C17.BENCH C432.BENCH C499.BENCH C880.BENCH C1355.BENCH และ C1980.BENCH ซึ่งผลการทำงานของ

เครื่องมือออกแบบวงจรแบบกราฟิก เครื่องมือสังเคราะห์วงจร และเครื่องมือจำลองการทำงานสามารถทำงานได้อย่างถูกต้อง ซึ่งสนับสนุนแนวคิดเชิงวัตถุทั้งสามแนวคิดที่นำมาใช้ในการออกแบบเครื่องมือ

เมื่อนำผลการทดลองที่ได้จากเครื่องมือออกแบบวงจรตรรกะเชิงผสมที่ได้พัฒนาขึ้นไปเปรียบเทียบกับเครื่องมือออกแบบวงจรอื่นที่นำมาใช้ในการออกแบบวงจรมาตรฐานเดียวกัน ปรากฏว่าค่าความจริงเอาต์พุตที่ได้ถูกต้องและตรงกัน ดังนั้นจึงสามารถสรุปได้ว่าเครื่องมือออกแบบวงจรตรรกะเชิงผสมที่สร้างขึ้นในวิทยานิพนธ์นี้น่าจะเป็นแนวทางและต้นแบบการออกแบบฮาร์ดแวร์โดยใช้กรรมวิธีของแนวคิดเชิงวัตถุมาประยุกต์กับงานฮาร์ดแวร์ได้อย่างดี

9.2 **ข้อเสนอแนะ**

เครื่องมือออกแบบวงจรตรรกะเชิงผสมแบบแนวคิดเชิงวัตถุที่นำเสนอในวิทยานิพนธ์นี้ แม้จะมีความสมบูรณ์ในแง่ที่สามารถทำงานให้ผลลัพธ์ออกมาถูกต้อง แต่การทำงานในเครื่องมือจำลองการทำงานยังให้ประสิทธิภาพที่ไม่ดีเนื่องจากใช้เวลาในการทำงานมาก ซึ่งอาจแก้ไขได้โดยสร้างเครื่องมือลดรูปวงจร (Optimizer) โดยใช้หลักการลดรูปสมการบูลีนเพื่อลดขนาดของวงจรให้มีความซับซ้อนน้อยลง จึงนำผลลัพธ์ของเครื่องมือลดรูปวงจรไปจำลองการทำงาน สำหรับขั้นตอนการพัฒนาเครื่องมือในวิทยานิพนธ์นี้ โปรแกรมไมโครซอฟท์ วิสิโอ ที่ได้นำมาใช้ในการพัฒนา ซึ่งประกอบด้วย ส่วนสเทนซิล รูปร่าง และส่วนของโปรแกรม มีข้อจำกัดทางด้านการพัฒนาเครื่องมือ คือ สเทนซิลและรูปร่าง มีความทนทานต่อการออกแบบต่ำ และส่วนของโปรแกรม วิชวลเบสิกสำหรับแอพลิเคชัน มีคำสั่งที่สนับสนุนการเขียนโปรแกรมโดยตรงไม่เพียงพอ จึงทำให้การพัฒนาเครื่องมือเสียแรงงานและเวลามาก เมื่อแบบแนวคิดเชิงวัตถุสามารถนำมาใช้ในการออกแบบวงจรตรรกะเชิงผสมได้อย่างถูกต้องแล้ว แบบแนวคิดเชิงวัตถุสามารถนำไปใช้เป็นประเด็นศึกษาเพิ่มเติมได้ คือ แบบแนวคิดเชิงวัตถุสำหรับออกแบบวงจรเชิงลำดับและเครื่องมือที่ได้พัฒนาขึ้นในวิทยานิพนธ์นี้ สามารถใช้เป็นต้นแบบในการนำไปพัฒนาให้มีประสิทธิภาพดียิ่งขึ้น

รายการอ้างอิง

1. Schumacher, G.; and Nebel W. Inheritance Concept for Signals in Object-Oriented Extensions to VHDL. European Design Automation Conference with EURO-VHDL '95 on EURO-DAC 1995 : 428-435.
2. Morris, D.; Evans, G.; and Green, P. An Integrated Approach to Engineering Computer Systems. European Design Automation Conference with EURO-VHDL '96 and Exhibition on European Design 1996 : 264-269.
3. Svarstad, K.; Nicolescu, G.; and Jerraya, A. A Model for Describing Communication between Aggregate Objects in the Specification and Design of Embedded Systems. Design, Automation, and Test in Europe 2001 : 77-85.
4. Edwards, M.; and Green, P. An Object Oriented Design Method for Reconfigurable Computing Systems. Design, Automation, and Test in Europe 2000 : 692-696.
5. Liao, S.; Tjiang, S.; and Gupta, R. An Efficient Implementation of Reactivity for Modeling Hardware in the Scenic Design Environment. Design Automation Conference 1997 : 70-75.
6. Ashenden, P.; and Wilsey, P. Considerations on Object-Oriented Extensions to VHDL. VIUF Conference 1997 : 109-118.
7. Kuhn, T.; Oppold, T.; Edwards, M.; and Kashai, Y. Object Oriented Hardware Synthesis and Verification. Design Automation Conference 2001 : 189-194.
8. Nebel, W.; and Schumacher G. Object-Oriented Hardware Modelling Where to apply and what are the Objects?. EURO-DAC '96 with EURO-VHDL '96 1996 : 428-433.
9. Smith, D. VHDL & Verilog Compared & Contrasted – Plus Modeled Example Written in VHDL, Verilog and C. Design Automation Conference 1996 : 33-39.
10. Maginot, S. Evaluation Criteria of HDLs : VHDL compared to Verilog, UDL/I & M. Design Automation Conference 1992 : 746-751.
11. Laquer, E. Object-Oriented System Engineering: A Method for Managing VHDL Development. VIUF Conference 1995 : 4.7-4.16.
12. Jantsch, A.; and Sander, I. On the Roles of Functions and Objects in System Specification. International Conference on Hardware Software Codesign 2000 :

8-12.

13. Kuhn, T.; Rosenstiel, W.; and Kebschull, U. Description and Simulation of Hardware/Software Systems with Java. ACM/IEEE Conference on Design Automation 1999 : 790-793.
14. Hutchings, B.; and Nelson, B. Using General-Purpose Programming Languages for FPGA Design. Conference on Design Automation 2000 : 561-566.
15. Jacome, M.; and Peixoto, H. A Survey of Digital Design Reuse. IEEE Design & Test of Computer 2001 : 98-107.
16. Budd, T. An Introduction to Object-Oriented Programming. 2nd ed. United States of America : Addison Wesley, 1997.
17. Priestle, M. Practical Object-Oriented Design with UML. Singapore : McGraw-Hill, 2000.
18. Pressman, R. Software Engineering A Practitioner's Approach. 4th ed. Singapore : McGraw-Hill, 1997.
19. Katz, R. Contemporary Logic Design. United States of America : The Benjamin/Cummings, 1994.
20. Booth, T. Introduction to Computer Engineering Hardware and Software Design. 3rd ed. United States of America : John Wiley & Sons, 1979.
21. Capilano Computing System Ltd. LogicWorks 4 INTERACTIVE CIRCUIT DESIGN SOFTWARE. United States of America : Addison Wesley, 1998.
22. DEVELOPING MICROSOFT VISIO SOLUTIONS. 1st ed. United States of America : Microsoft Press, 2001.
23. Bening, L.; and Foster, H. Principles of Verifiable RTL Design A Functional Coding Style Supporting Verification Processes in Verilog. United States of America : Kluwer Academic, 2000.
24. International Symposium Circuit And Systems 1985[Online]. Available from: http://www.cbl.ncsu.edu/www/CBL_Docs/iscas85.htm.

ภาคผนวก

ภาคผนวก ก
วิธีการใช้เครื่องมือออกแบบ
วงจรตรรกะเชิงผสมแบบแนวคิดเชิงวัตถุ

เครื่องมือออกแบบวงจรตรรกะเชิงผสมแบบแนวคิดเชิงวัตถุที่พัฒนาขึ้นในวิทยานิพนธ์นี้ ประกอบด้วยเครื่องมือทั้งหมดสามเครื่องมือ ซึ่งพัฒนาขึ้นภายในโปรแกรมไมโครซอฟท์ วิสิโอ เวอร์ชัน Enterprise Architect เนื้อหาในส่วนนี้จะอธิบายถึงวิธีการใช้เครื่องมือออกแบบวงจรตรรกะเชิงผสมแบบแนวคิดเชิงวัตถุที่พัฒนาขึ้น ซึ่งแบ่งการอธิบายออกเป็นห้าส่วน คือ (1) การติดตั้งเครื่องมือที่พัฒนาขึ้นในโปรแกรม (2) การเรียกใช้งานโปรแกรม (3) การใช้งานส่วนออกแบบวงจรแบบกราฟิก (4) การใช้งานเครื่องมือสังเคราะห์วงจร และ (5) การใช้งานเครื่องมือจำลองการทำงาน

ก.1 การติดตั้งเครื่องมือที่พัฒนาขึ้นในโปรแกรม

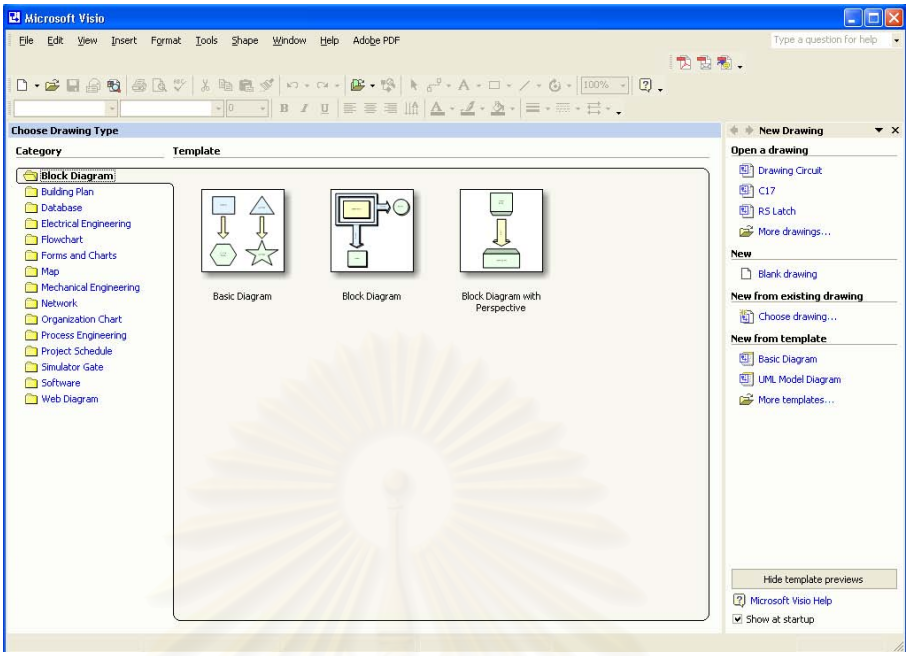
เครื่องมือออกแบบวงจรตรรกะเชิงผสมแบบแนวคิดเชิงวัตถุที่พัฒนาขึ้นในวิทยานิพนธ์นี้ เนื่องจากเป็นเครื่องมือที่สร้างขึ้นมาเฉพาะเจาะจงตามวัตถุประสงค์การใช้งาน ภายใต้โปรแกรมไมโครซอฟท์ วิสิโอ ดังนั้นผู้ออกแบบฮาร์ดแวร์จำเป็นต้องติดตั้งเครื่องมือที่พัฒนาขึ้นลงในโปรแกรมวิสิโอ ซึ่งมีขั้นตอนทั้งหมดสองขั้นตอน คือ

1. คัดลอกโฟลเดอร์ชื่อ Simulator Gate ซึ่งเก็บไฟล์ของสเทนซิลที่ใช้สำหรับการออกแบบวงจรลงใน C:\Program Files\Microsoft Office\Visio10\1033\Solutions
2. คัดลอกไฟล์ชื่อ Drawing Circuit ซึ่งเป็นไฟล์เครื่องมือออกแบบวงจรตรรกะเชิงผสมที่ถูกพัฒนาขึ้นไปยังตำแหน่งที่ต้องการ

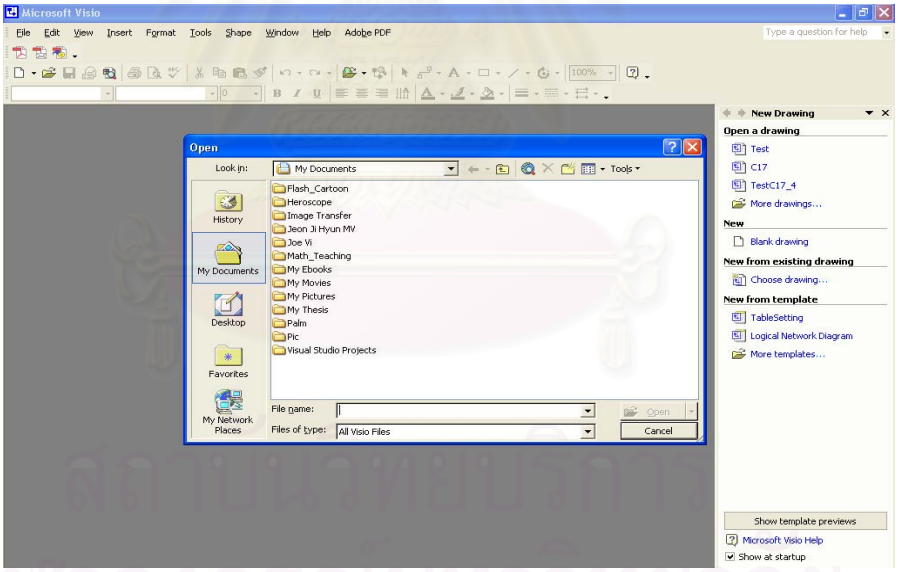
ก.2 การเรียกใช้งานเครื่องมือออกแบบวงจรตรรกะเชิงผสม

เครื่องมือออกแบบวงจรตรรกะเชิงผสมแบบแนวคิดเชิงวัตถุที่ถูกพัฒนาขึ้น มีขั้นตอนการเรียกใช้งานสามขั้นตอน คือ

1. เปิดโปรแกรมไมโครซอฟท์ วิสิโอ ดังแสดงในรูปที่ ก.1
2. เลือกปุ่ม  หรือเลือก File -> Open เมื่อผู้ออกแบบฮาร์ดแวร์เลือกแล้วจะปรากฏหน้าต่างโต้ตอบดังรูปที่ ก.2 และให้ผู้ออกแบบฮาร์ดแวร์เปิดไฟล์ชื่อ Drawing Circuit ซึ่งอยู่ในไดเรกทอรีที่ผู้ออกแบบฮาร์ดแวร์กำหนดไว้ในส่วน ก.1



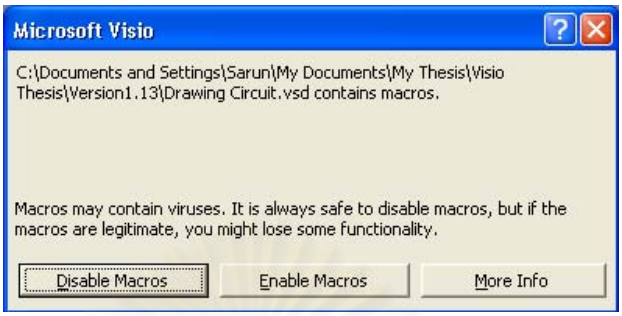
รูปที่ ก.1 หน้าต่างเริ่มต้นของโปรแกรม ไมโครซอฟท์ วิสิโอ



รูปที่ ก.2 หน้าต่างโต้ตอบเพื่อเปิดไฟล์ที่ต้องการใช้งาน

3. หลังจากที่ผู้ออกแบบฮาร์ดแวร์เปิดไฟล์แล้ว จะปรากฏหน้าต่างโต้ตอบสำหรับการตัดสินใจกำหนดให้แมโคร (Macro) ทำงานหรือไม่ ดังแสดงในรูปที่ ก.3 ถ้าผู้ออกแบบฮาร์ดแวร์เลือกปุ่ม Enable Macros เครื่องมือออกแบบวงจรตรรกะเชิงผสมที่ได้พัฒนาขึ้นจะทำงานได้อย่างสมบูรณ์ แต่ถ้าผู้ออกแบบฮาร์ดแวร์เลือกปุ่ม Disable Macros

เครื่องมือออกแบบวงจรตรรกะเชิงผสมที่ได้พัฒนาขึ้นจะไม่สามารถทำงานได้อย่างสมบูรณ์



รูปที่ ก.3 หน้าต่างโต้ตอบเพื่อเลือกการทำงานแมโคร

ก.3 การใช้งานส่วนออกแบบวงจรแบบกราฟิก

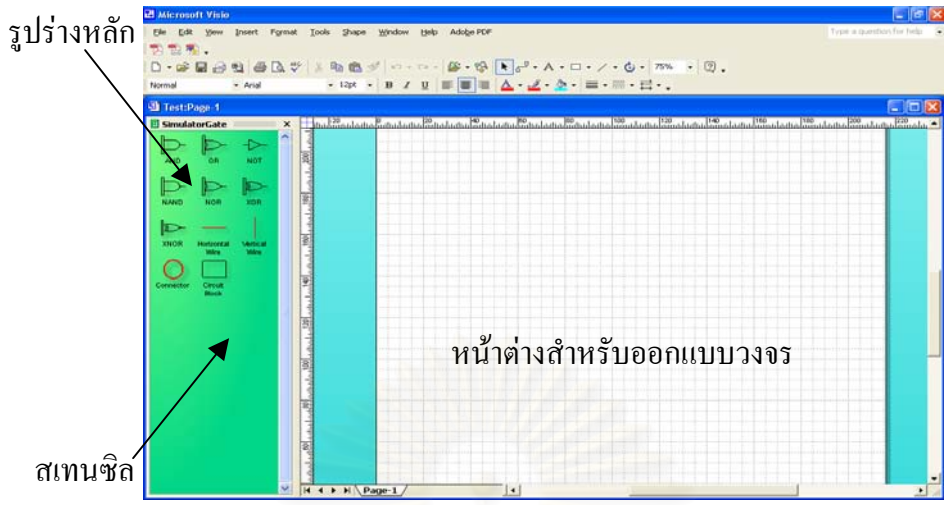
การใช้งานส่วนออกแบบวงจรแบบกราฟิก สามารถแบ่งการอธิบายออกได้เป็นสี่ส่วน คือ 1) การใช้เกทสำหรับออกแบบวงจร 2) การใช้สายสัญญาณสำหรับออกแบบวงจร 3) การใช้จุดเชื่อมต่อสำหรับออกแบบวงจร และ 4) การใช้วงจรบล็อกสำหรับออกแบบวงจร

ก.3.1 การใช้เกทสำหรับออกแบบวงจร

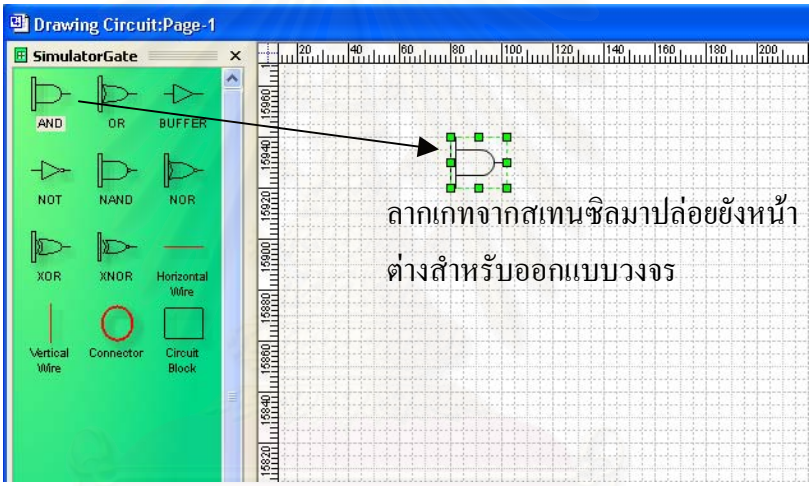
รูป ก.4 แสดงส่วนออกแบบวงจรแบบกราฟิก ซึ่งประกอบด้วย รูปแบบหลักของอุปกรณ์สำหรับออกแบบวงจรตรรกะเชิงผสมซึ่งอยู่ภายในสเทนซิล และหน้าต่างออกแบบวงจรตรรกะเชิงผสม รูปแบบหลักของเกทที่ถูกพัฒนาขึ้นในวิทยานิพนธ์นี้จำนวนแปดรูปร่างหลัก คือ แอนด์ ออร์ บัฟเฟอร์ ตัวผกผัน แนนด์ นอร์ ออร์เฉพาะ และออร์ไม่เฉพาะ ผู้ออกแบบฮาร์ดแวร์สามารถนำเกทดังกล่าวมาใช้ในการออกแบบวงจรโดยใช้หลักการลากเกทที่ต้องการใช้ออกแบบวงจรจากสเทนซิลไปปล่อยยังหน้าต่างสำหรับออกแบบวงจร ดังแสดงในรูปที่ ก.5

ก.3.2 การใช้สายสัญญาณสำหรับออกแบบวงจร

การใช้สายสัญญาณสำหรับออกแบบวงจรตรรกะเชิงผสมสำหรับเครื่องมือที่พัฒนาขึ้นนี้สามารถแบ่งอธิบายการใช้สายสัญญาณออกได้เป็นสามรูปแบบ คือ 1) การใช้สายสัญญาณเพื่อเป็นอินพุตหรือเอาต์พุตหลักเมื่อเชื่อมต่อกับเกท 2) การใช้สายสัญญาณเพื่อเชื่อมระหว่างสายสัญญาณกับสายสัญญาณ และ 3) การใช้สายสัญญาณเพื่อเป็นอินพุตหรือเอาต์พุตใดๆ เมื่อเชื่อมต่อกับเกท

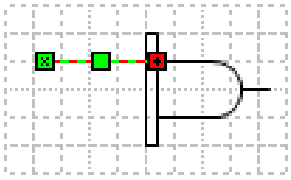


รูปที่ ก.4 ส่วนออกแบบวงจรแบบกราฟิก



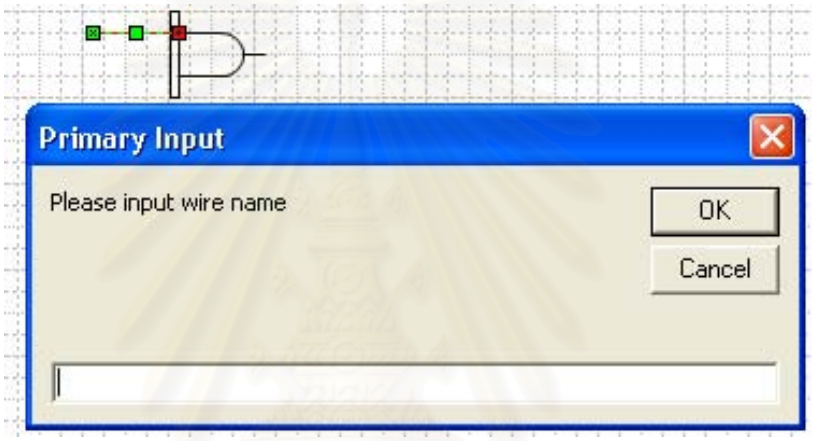
รูปที่ ก.5 การใช้เกทสำหรับออกแบบวงจรด้วยวิธีการลากแล้วปล่อย

ก.3.2.1 การใช้สายสัญญาณเพื่อเป็นอินพุตหรือเอาต์พุตหลักเมื่อเชื่อมต่อกับเกท รูปร่างหลักของสายสัญญาณที่ใช้สำหรับออกแบบวงจรภายในสเทน-ซิล คือ Horizontal Wire และ Vertical Wire เมื่อผู้ออกแบบฮาร์ดแวร์ต้องการกำหนดให้สายสัญญาณชนิดที่ต้องการเป็นอินพุตหลัก สามารถทำได้โดยลากสายสัญญาณชนิดที่ต้องการใช้ออกแบบวงจรจากสเทนซิลไปปล่อยยังส่วนรับอินพุตของเกทที่ต้องการเชื่อมต่อ โดยที่จุดปลายของสายสัญญาณต้องอยู่ในตำแหน่งส่วนรับอินพุตของเกท ดังแสดงในรูปที่ ก.6



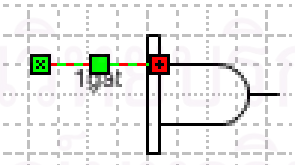
รูปที่ ก.6 การใช้สายสัญญาณเพื่อเป็นอินพุตหลักเมื่อเชื่อมต่อกับเกท

เมื่อจุดปลายของสายสัญญาณถูกเชื่อมต่อเข้ากับส่วนรับอินพุตของเกทแล้วจะปรากฏหน้าต่างโต้ตอบสำหรับกำหนดชื่อของสายสัญญาณที่เป็นอินพุตหลัก ดังแสดงในรูปที่ ก.7



รูปที่ ก.7 หน้าต่างโต้ตอบสำหรับกำหนดชื่อของสายสัญญาณอินพุตหลัก

เมื่อผู้ออกแบบฮาร์ดแวร์กำหนดชื่อสายสัญญาณสำหรับอินพุตหลักแล้ว ชื่อของสายสัญญาณดังกล่าวจะปรากฏใต้ตำแหน่งของสายสัญญาณนั้นในหน้าต่างการออกแบบวงจร ดังแสดงในรูปที่ ก.8

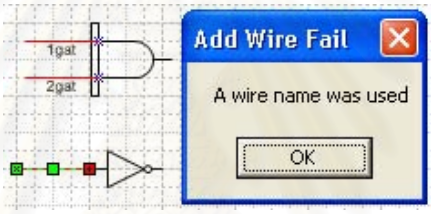


รูปที่ ก.8 ชื่อของสายสัญญาณที่กำหนดปรากฏในหน้าต่างการออกแบบ

แต่ถ้าผู้ออกแบบฮาร์ดแวร์มิได้กำหนดชื่อของสายสัญญาณแล้วกดปุ่ม OK จะปรากฏหน้าต่างโต้ตอบเพื่อเตือนผู้ออกแบบฮาร์ดแวร์ให้กำหนดชื่อของสายสัญญาณ ดังแสดงในรูปที่ ก.9

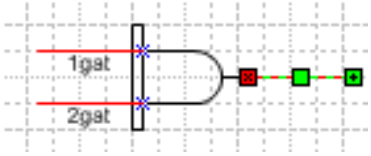
รูปที่ ก.9 หน้าต่างโต้ตอบเพื่อเตือนให้ผู้ออกแบบฮาร์ดแวร์กำหนดชื่อของสายสัญญาณ

แต่ถ้าชื่อของสายสัญญาณที่ผู้ออกแบบฮาร์ดแวร์กำหนดแล้วซ้ำกับชื่อของอินพุตหลักหรือเอาต์พุตที่ได้ออกแบบไว้แล้ว เครื่องมือจะแสดงหน้าต่างโต้ตอบเพื่อเตือนให้ผู้ออกแบบฮาร์ดแวร์กำหนดชื่อของอินพุตหลักอีกครั้งหนึ่ง ดังแสดงในรูปที่ ก.10



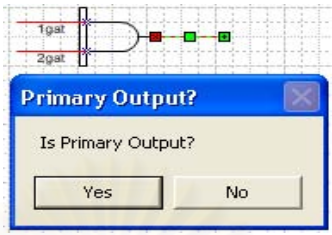
รูปที่ ก.10 หน้าต่างโต้ตอบเพื่อเตือนการกำหนดชื่อซ้ำจากผู้ออกแบบฮาร์ดแวร์

แต่ถ้าผู้ออกแบบฮาร์ดแวร์ต้องการกำหนดให้สายสัญญาณที่ต้องการเป็นเอาต์พุตหลัก สามารถทำได้โดยลากสายสัญญาณชนิดที่ต้องการใช้ออกแบบวงจรจากสเทนซิลไปปล่อยยังส่วนให้เอาต์พุตของเกทที่ต้องการเชื่อมต่อ โดยที่จุดเริ่มต้นของสายสัญญาณต้องอยู่ในตำแหน่งส่วนให้เอาต์พุตของเกท และผู้ออกแบบฮาร์ดแวร์จะกำหนดสายสัญญาณที่เป็นเอาต์พุตหลักของเกทได้ก็ต่อเมื่อเกทดังกล่าวได้มีกำหนดสายสัญญาณอินพุตที่เป็นอินพุตหลักหรืออินพุตใดๆ เรียบร้อยแล้ว ดังแสดงในรูปที่ ก.11



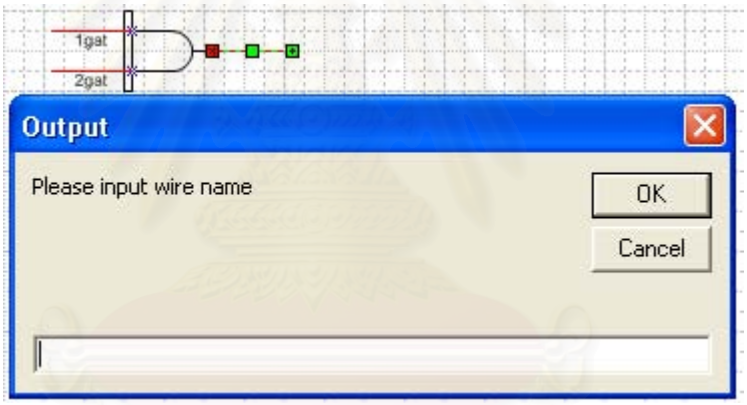
รูปที่ ก.11 การเชื่อมต่อสายสัญญาณเข้ากับส่วนให้เอาต์พุตของเกท

เมื่อสายสัญญาณได้เชื่อมต่อเข้ากับส่วนให้เอาต์พุตของเกทแล้วจะปรากฏหน้าต่างโต้ตอบเพื่อให้ผู้ออกแบบฮาร์ดแวร์เลือกว่าเอาต์พุตดังกล่าวเป็นเอาต์พุตหลักหรือไม่ ดังแสดงในรูปที่ ก.12



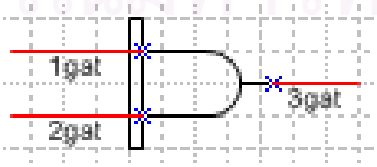
รูปที่ ก.12 หน้าต่างโต้ตอบสำหรับกำหนดเอาต์พุตหลัก

เมื่อผู้ออกแบบฮาร์ดแวร์เลือกปุ่ม Yes เพื่อกำหนดให้สายสัญญาณดังกล่าวเป็นเอาต์พุตหลักแล้ว หน้าต่างโต้ตอบสำหรับกำหนดชื่อของสายสัญญาณที่เป็นเอาต์พุตหลักจะปรากฏขึ้น ดังแสดงในรูปที่ ก.13



รูปที่ ก.13 หน้าต่างโต้ตอบสำหรับกำหนดชื่อของสายสัญญาณเอาต์พุตหลัก

เมื่อผู้ออกแบบฮาร์ดแวร์กำหนดชื่อสายสัญญาณสำหรับเอาต์พุตหลักแล้ว ชื่อของสายสัญญาณดังกล่าวจะปรากฏใต้ตำแหน่งของสายสัญญาณนั้นในหน้าต่างการออกแบบวงจร ดังแสดงในรูปที่ ก.14

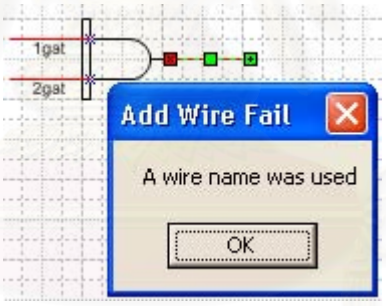


รูปที่ ก.14 ชื่อของสายสัญญาณที่กำหนดปรากฏในหน้าต่างการออกแบบ

แต่ถ้าผู้ออกแบบฮาร์ดแวร์มิได้กำหนดชื่อของสายสัญญาณแล้วกดปุ่ม OK จะปรากฏหน้าต่างโต้ตอบเพื่อเตือนผู้ออกแบบฮาร์ดแวร์ให้กำหนดชื่อของสายสัญญาณ ดังแสดงในรูปที่ ก.15

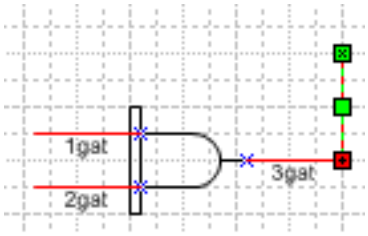
รูปที่ ก.15 หน้าต่างโต้ตอบเพื่อเตือนให้ผู้ออกแบบฮาร์ดแวร์กำหนดชื่อของสายสัญญาณ

แต่ถ้าชื่อของสายสัญญาณที่ผู้ออกแบบฮาร์ดแวร์กำหนดแล้วซ้ำกับชื่อของอินพุตหลักหรือเอาต์พุตที่ได้ออกแบบไว้แล้ว เครื่องมือจะแสดงหน้าต่างโต้ตอบเพื่อเตือนให้ผู้ออกแบบฮาร์ดแวร์กำหนดชื่อของเอาต์พุตหลักอีกครั้งหนึ่ง ดังแสดงในรูปที่ ก.16



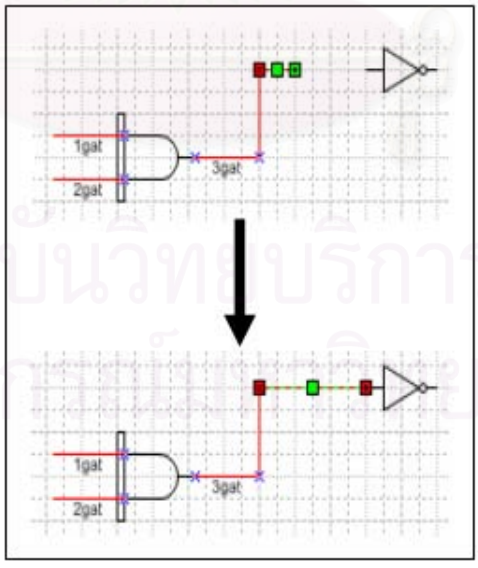
รูปที่ ก.16 หน้าต่างโต้ตอบเพื่อเตือนการกำหนดชื่อซ้ำจากผู้ออกแบบฮาร์ดแวร์

ก.3.2.2 การใช้สายสัญญาณเพื่อเชื่อมระหว่างสายสัญญาณกับสายสัญญาณ สายสัญญาณที่ถูกออกแบบไว้ในวงจรสามารถขยาย หด หรือ เปลี่ยนทิศทางได้โดยลากที่จุดปลายของสายสัญญาณที่ออกแบบไว้ไปยังทิศที่ต้องการ แต่การเปลี่ยนทิศทางอาจไม่สามารถกระทำได้ในบางครั้ง จึงจำเป็นต้องใช้วิธีการเชื่อมต่อสายสัญญาณ ซึ่งการเชื่อมต่อสายสัญญาณจะคงค่าความจริงของสัญญาณที่ได้ทำการเชื่อมต่อไว้ การเชื่อมต่อสายสัญญาณทำได้โดย นำจุดเริ่มต้นของสายสัญญาณใหม่ที่ต้องการเชื่อมต่อมาเชื่อมเข้ากับจุดปลายของสายสัญญาณเดิม ดังแสดงในรูปที่ ก.17



รูปที่ ก.17 ตัวอย่างการเชื่อมต่อสายสัญญาณเข้ากับสายสัญญาณ

ก.3.2.3 การใช้สายสัญญาณเพื่อเป็นอินพุตหรือเอาต์พุตใดๆ เมื่อเชื่อมต่อกับเกท สายสัญญาณที่เป็นอินพุตใดๆ ที่มิใช่อินพุตหลักนั้นจำเป็นต้องเกิดมาจากเอาต์พุตใดๆ ดังนั้นการใช้สายสัญญาณเพื่อเป็นอินพุตใดๆ นั้นจะทำได้โดยการเชื่อมต่อสายสัญญาณจากเอาต์พุตใดๆ ไปยังส่วนรับอินพุตของเกทที่ต้องการ ดังแสดงตัวอย่างในรูปที่ ก.18 โดยผู้ออกแบบฮาร์ดแวร์ห้ามทำการเชื่อมสายสัญญาณเข้ากับเกทก่อนที่จะลากสายสัญญาณมาเชื่อมเข้ากับสายสัญญาณเอาต์พุตใดๆ เพราะการเชื่อมสายสัญญาณเข้ากับเกทก่อนจะใช้สำหรับกำหนดให้สายสัญญาณดังกล่าวทำหน้าที่เป็นอินพุตหลัก และเมื่อผู้ออกแบบฮาร์ดแวร์ออกแบบเกทใดๆ ผู้ออกแบบฮาร์ดแวร์จำเป็นต้องออกแบบเกทที่กำลังออกแบบให้เสร็จจึงสามารถไปออกแบบเกทอื่นๆ ต่อได้

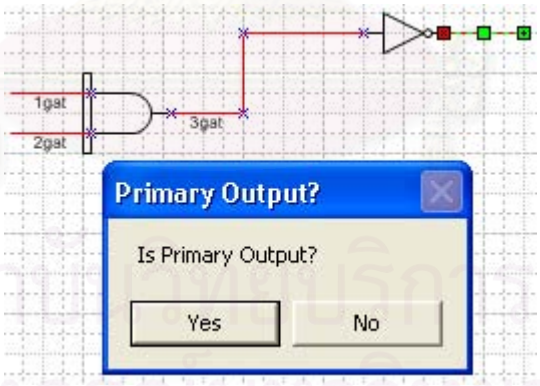


รูปที่ ก.18 ตัวอย่างการเชื่อมต่อสายสัญญาณเพื่อเป็นอินพุตใดๆ

แต่ถ้าผู้ออกแบบฮาร์ดแวร์ต้องการกำหนดให้สายสัญญาณที่ต้องการเป็นเอาต์พุตใดๆสามารถทำได้โดยลากสายสัญญาณชนิดที่ต้องการใช้ออกแบบวงจรจากสเทนซิลไปปล่อยยังส่วนให้เอาต์พุตของเกทที่ต้องการเชื่อมต่อ โดยที่จุดเริ่มต้นของสายสัญญาณต้องอยู่ในตำแหน่งส่วนให้เอาต์พุตของเกท และผู้ออกแบบฮาร์ดแวร์จะกำหนดสายสัญญาณที่เป็นเอาต์พุตใดๆ ของเกทได้ก็ต่อเมื่อเกทดังกล่าวได้มีกำหนดสายสัญญาณอินพุตที่เป็นอินพุตหลักหรืออินพุตใดๆ เรียบร้อยแล้ว ดังแสดงในรูปที่ ก.19

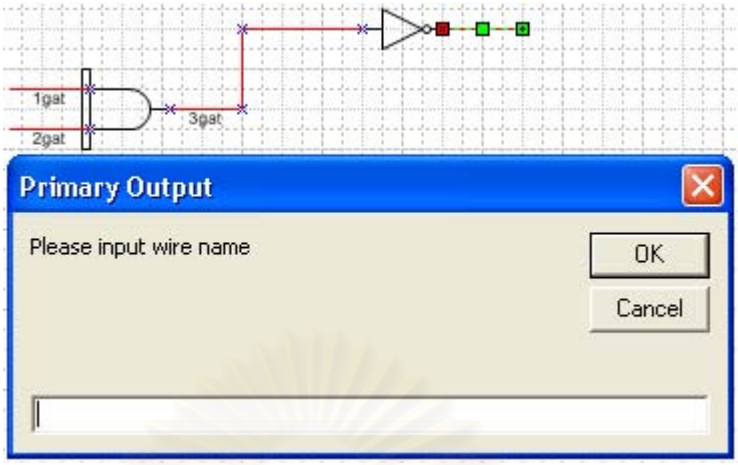
รูปที่ ก.19 การเชื่อมต่อสายสัญญาณเข้ากับส่วนให้เอาต์พุตของเกท

เมื่อสายสัญญาณได้เชื่อมต่อเข้ากับส่วนให้เอาต์พุตของเกทแล้วจะปรากฏหน้าต่างโต้ตอบเพื่อให้ผู้ออกแบบฮาร์ดแวร์เลือกว่าเอาต์พุตดังกล่าวเป็นเอาต์พุตหลักหรือไม่ ดังแสดงในรูปที่ ก.20



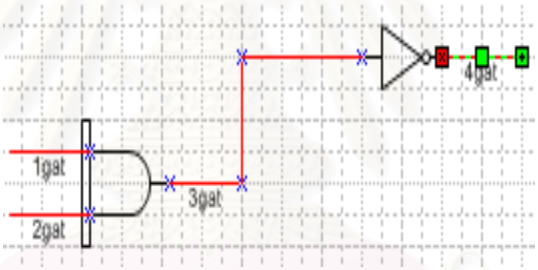
รูปที่ ก.20 หน้าต่างโต้ตอบสำหรับกำหนดเอาต์พุตหลักหรือเอาต์พุตใดๆ

เมื่อผู้ออกแบบฮาร์ดแวร์เลือกปุ่ม No เพื่อกำหนดให้สายสัญญาณดังกล่าวเป็นเอาต์พุตใดๆ แล้ว หน้าต่างโต้ตอบสำหรับกำหนดชื่อของสายสัญญาณที่เป็นเอาต์พุตใดๆ จะปรากฏขึ้น ดังแสดงในรูปที่ ก.21



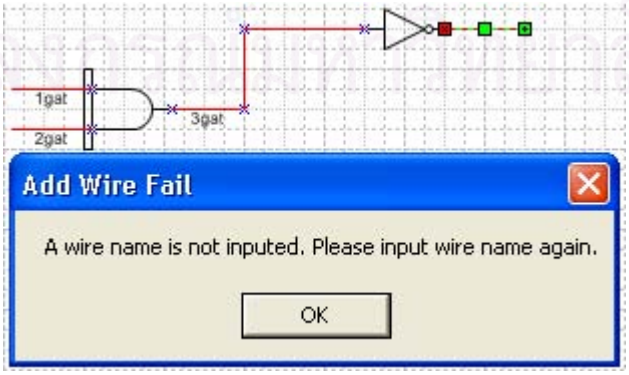
รูปที่ ก.21 หน้าต่างโต้ตอบสำหรับกำหนดชื่อของสายสัญญาณเอาต์พุตใดๆ

เมื่อผู้ออกแบบฮาร์ดแวร์กำหนดชื่อสายสัญญาณสำหรับเอาต์พุตใดๆ แล้ว ชื่อของสายสัญญาณดังกล่าวจะปรากฏใต้ตำแหน่งของสายสัญญาณนั้นในหน้าต่างการออกแบบวงจร ดังแสดงในรูปที่ ก.22



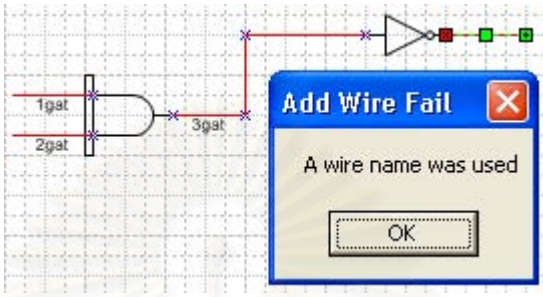
รูปที่ ก.22 ชื่อของสายสัญญาณที่กำหนดปรากฏในหน้าต่างการออกแบบ

แต่ถ้าผู้ออกแบบฮาร์ดแวร์ไม่ได้กำหนดชื่อของสายสัญญาณแล้วกดปุ่ม OK จะปรากฏหน้าต่างโต้ตอบเพื่อเตือนผู้ออกแบบฮาร์ดแวร์ให้กำหนดชื่อของสายสัญญาณ ดังแสดงในรูปที่ ก.23



รูปที่ ก.23 หน้าต่างโต้ตอบเพื่อเตือนให้ผู้ออกแบบฮาร์ดแวร์กำหนดชื่อของสายสัญญาณ

แต่ถ้าชื่อของสายสัญญาณที่ผู้ออกแบบฮาร์ดแวร์กำหนดแล้วซ้ำกับชื่อของอินพุตหลักหรือเอาต์พุตที่ได้ออกแบบไว้แล้ว เครื่องมือจะแสดงหน้าต่างโต้ตอบเพื่อเตือนให้ผู้ออกแบบฮาร์ดแวร์กำหนดชื่อของเอาต์พุตใดๆ อีกครั้งหนึ่ง ดังแสดงในรูปที่ ก.24



รูปที่ ก.24 หน้าต่างโต้ตอบเพื่อเตือนการกำหนดชื่อซ้ำจากผู้ออกแบบฮาร์ดแวร์

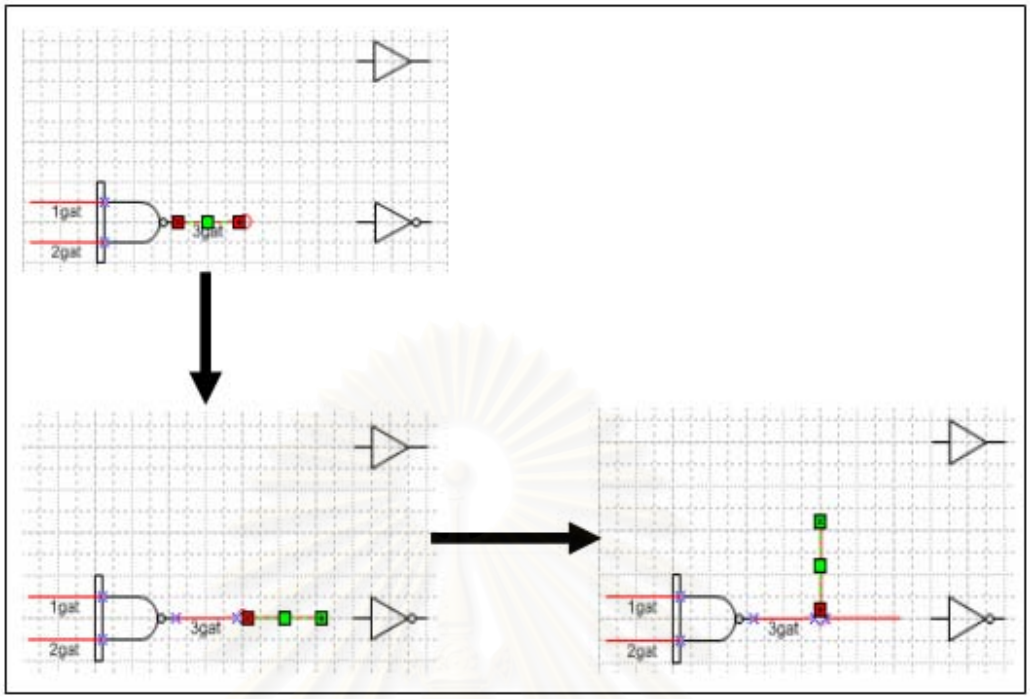
ก.3.3 การใช้จุดเชื่อมต่อสำหรับออกแบบวงจร

จุดเชื่อมต่อใช้ในการออกแบบวงจรเมื่อสายสัญญาณใดๆ ที่ถูกออกแบบในวงจรถูกแยกออกเป็นอินพุตของเกทที่มีจำนวนมากกว่าหนึ่งเกท รูปร่างหลักของจุดเชื่อมต่อภายในสเทนซิล คือ Connector การใช้งานจุดเชื่อมต่อสำหรับเครื่องมือออกแบบวงจรที่ได้พัฒนาขึ้นนี้ ทำได้โดยการลากจุดปลายของสายสัญญาณที่ต้องการแยกออกไปเป็นอินพุตของเกทอีกเกทหนึ่งซึ่งได้ออกแบบไว้แล้วไปเชื่อมต่อเข้ากับจุดเชื่อมต่อที่ลากจากสเทนซิลมาปล่อยยังหน้าต่างการออกแบบ หลังจากนั้นให้นำจุดเริ่มต้นของสายสัญญาณมาเชื่อมต่อเข้ากับจุดเชื่อมต่อในทิศทางที่ผู้ออกแบบฮาร์ดแวร์ต้องการออกแบบ ดังแสดงขั้นตอนการใช้งานจุดเชื่อมต่อในรูปที่ ก.25

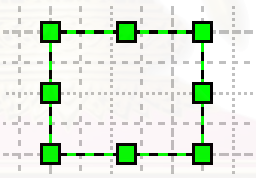
ก.3.4 การใช้วงจรบล็อกสำหรับออกแบบวงจร

วงจรบล็อก คือ วงจรที่ถูกออกแบบและสังเคราะห์วงจรไว้แล้ว ซึ่งผู้ออกแบบฮาร์ดแวร์สามารถนำวงจรดังกล่าวมาใช้ในการออกแบบวงจรที่มีวงจรบล็อกเป็นส่วนประกอบได้ โดยที่ผู้ออกแบบฮาร์ดแวร์ไม่จำเป็นต้องออกแบบวงจรบล็อกใหม่อีกครั้งหนึ่ง ซึ่งสนับสนุนแนวคิดการนำกลับมาใช้ของแนวคิดเชิงวัตถุ สำหรับวงจรบล็อกมีวิธีการใช้งานในเครื่องมือที่ได้พัฒนาขึ้น โดยเริ่มต้นจากลากรูปร่างหลักที่มีชื่อว่า Circuit Block มาปล่อยลงในหน้าต่างของการออกแบบ ดังแสดงในรูปที่ ก.26

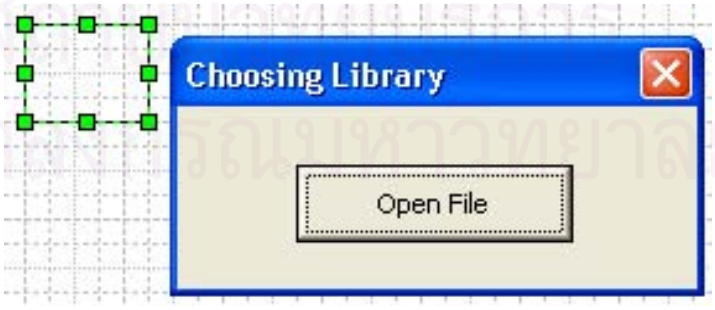
เมื่อวงจรบล็อกถูกปล่อยลงมายังหน้าต่างการออกแบบแล้ว จะปรากฏหน้าต่างโต้ตอบเพื่อให้ผู้ออกแบบฮาร์ดแวร์กดปุ่ม Open File ดังแสดงในรูปที่ ก.27 เพื่อไปยังหน้าต่างโต้ตอบของการเรียกใช้วงจรบล็อกซึ่งอยู่ในรูปแบบของไฟล์ข้อความ ดังแสดงในรูปที่ ก.28



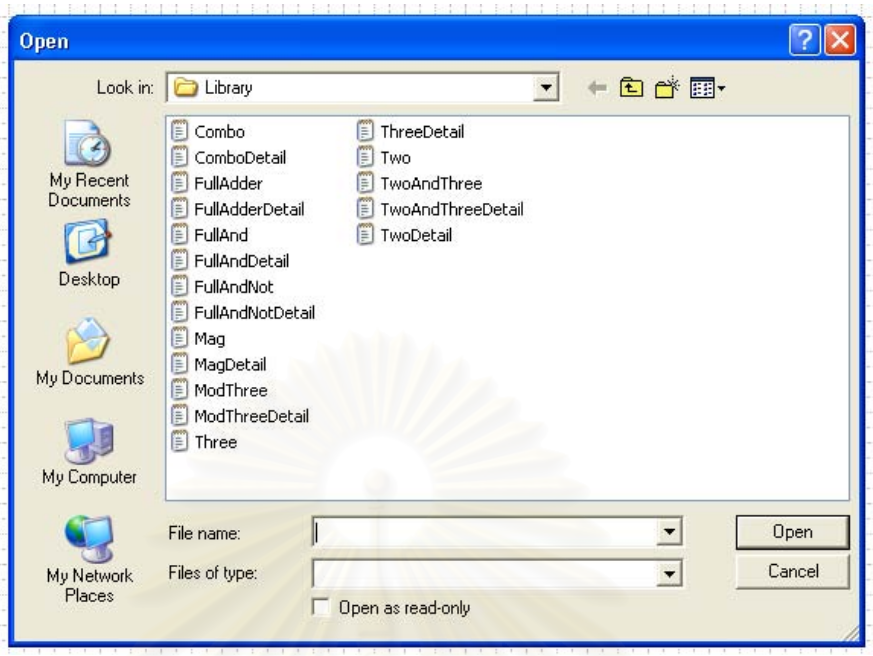
รูปที่ ก.25 ตัวอย่างการใช้จุดเชื่อมต่อสำหรับออกแบบวงจร



รูปที่ ก.26 การปล่อยวงจรบล็อกลงในหน้าต่างการออกแบบ

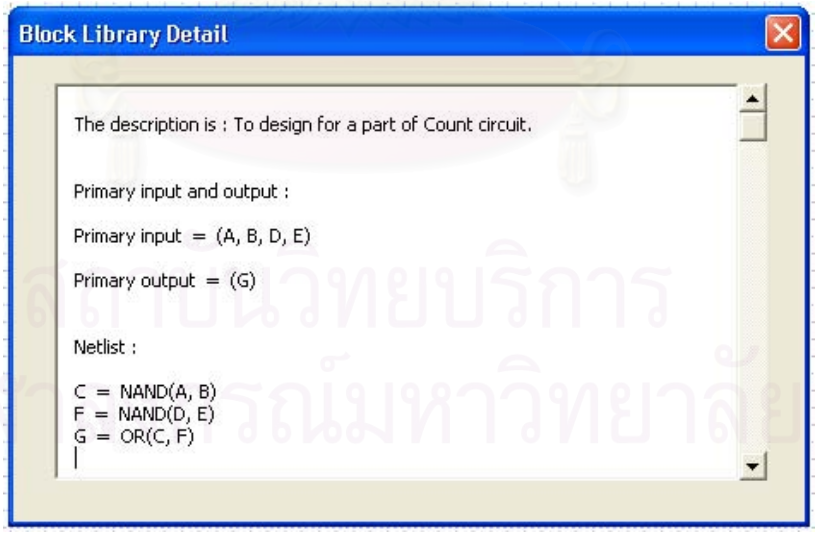


รูปที่ ก.27 หน้าต่างโต้ตอบเพื่อไปยังหน้าต่างเลือกวงจรบล็อก



รูปที่ ก.28 หน้าต่างโต้ตอบเพื่อเลือกวงจรบล็อกสำหรับการออกแบบ

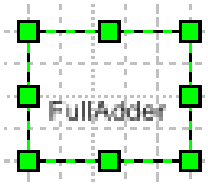
เมื่อผู้ออกแบบฮาร์ดแวร์เลือกไฟล์ของวงจรบล็อกที่ต้องการใช้ในการออกแบบแล้ว เครื่องมือออกแบบวงจรตรรกะเชิงผสมจะแสดงหน้าต่างโต้ตอบที่แสดงรายละเอียดของวงจรบล็อก ซึ่งประกอบไปด้วย 1) รายชื่อของอินพุตหลักของวงจรบล็อก 2) รายชื่อของเอาต์พุตหลักของวงจรบล็อก และ 3) ภาษาเนทลิสต์ของวงจรบล็อกที่ได้จากการสังเคราะห์วงจร ดังแสดงในรูปที่ ก.29



รูปที่ ก.29 รายละเอียดของวงจรบล็อกที่ผู้ออกแบบฮาร์ดแวร์เลือกสำหรับการออกแบบ

เมื่อผู้ออกแบบฮาร์ดแวร์ปิดหน้าต่างแสดงรายละเอียดของวงจรบล็อกที่เลือกแล้ว ชื่อของวงจรบล็อกที่เลือกจะปรากฏอยู่ภายในวงจรบล็อกที่ถูกนำมาปล่อยยังหน้าต่างการออกแบบ ดังแสดง

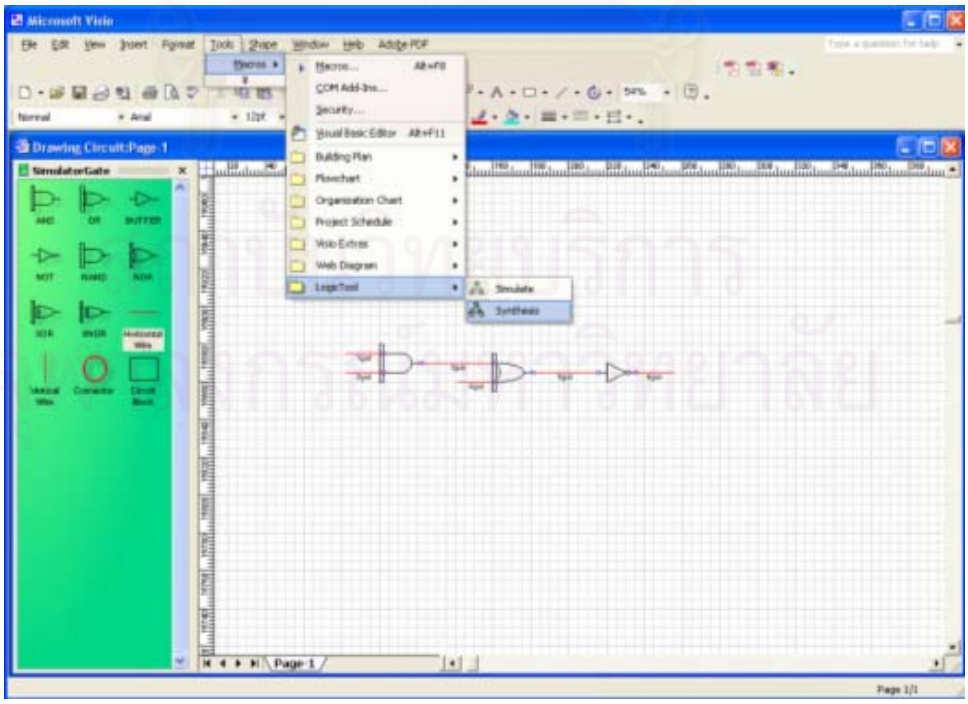
ในรูปที่ ก.30 ซึ่งวงจรบล็อกที่ได้ปรากฏบนหน้าต่างการออกแบบนี้พร้อมที่จะให้ผู้ออกแบบฮาร์ดแวร์นำไปใช้ในการออกแบบวงจรต่อไปได้



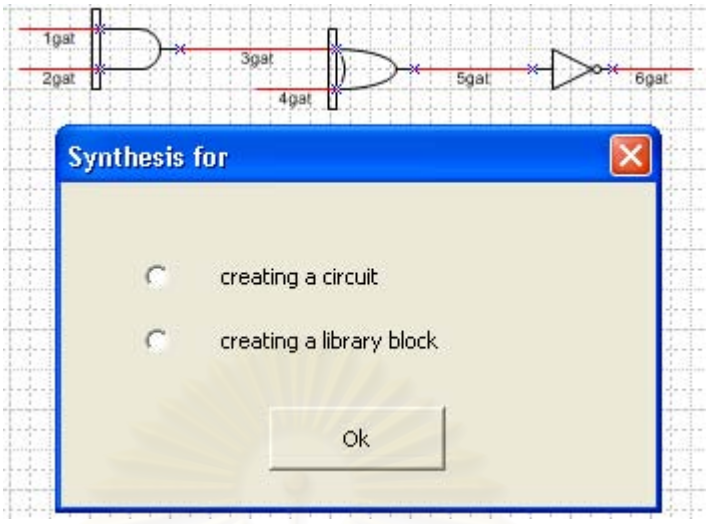
รูปที่ ก.30 ชื่อของวงจรบล็อกที่ได้เรียกใช้ภายในวงจรบล็อก

ก.4 การใช้งานเครื่องมือสังเคราะห์วงจร

เครื่องมือสังเคราะห์วงจรที่ได้พัฒนาขึ้น จะสามารถเรียกทำงานได้ก็ต่อเมื่อผู้ออกแบบฮาร์ดแวร์ได้ออกแบบวงจรตรรกะเชิงผสมเสร็จเป็นที่เรียบร้อยแล้ว ซึ่งผู้ออกแบบฮาร์ดแวร์สามารถเรียกใช้เครื่องมือสังเคราะห์วงจร โดยเรียก Tools -> Macros -> Logic Tool -> Synthesis ที่ตำแหน่งเมนูบาร์ ดังแสดงในรูปที่ ก.31 เมื่อผู้ออกแบบฮาร์ดแวร์เรียกเครื่องมือสังเคราะห์วงจรเพื่อใช้งาน เครื่องมือสังเคราะห์วงจรจะแสดงหน้าต่างโต้ตอบเพื่อให้ผู้ออกแบบฮาร์ดแวร์กำหนดจุดประสงค์ของการสังเคราะห์วงจรซึ่งประกอบไปด้วยสองวัตถุประสงค์ด้วยกัน คือ 1) เพื่อใช้สำหรับจำลองการทำงานโดยตรง และ 2) เพื่อใช้สำหรับสร้างเป็นวงจรบล็อก ซึ่งแสดงหน้าต่างโต้ตอบดังกล่าวในรูปที่ ก.32



รูปที่ ก.31 การเรียกใช้เครื่องมือสังเคราะห์วงจร

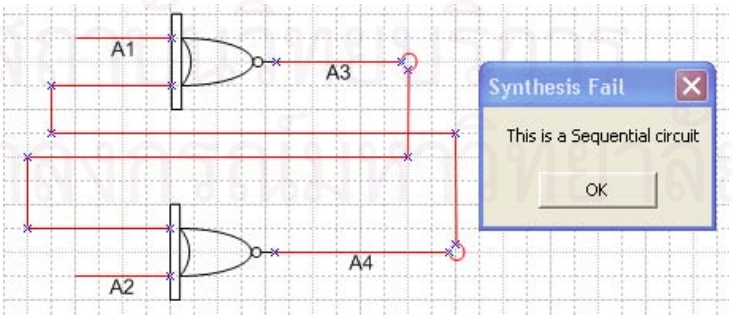


รูปที่ ก.32 หน้าต่างโต้ตอบเพื่อกำหนดจุดประสงค์ของการสังเคราะห์วงจร

เนื่องจากวัตถุประสงค์ของการสังเคราะห์วงจรมีสองวัตถุประสงค์ จึงอธิบายการใช้งานของแต่ละวัตถุประสงค์ ดังนี้

ก.4.1 การสังเคราะห์วงจรเพื่อจำลองการทำงานโดยตรง

เมื่อผู้ออกแบบฮาร์ดแวร์เลือก Creating a circuit เมื่อสังเคราะห์วงจรเพื่อจำลองการทำงานโดยตรงแล้ว เครื่องมือสังเคราะห์วงจรจะสังเคราะห์วงจรให้กับผู้ออกแบบฮาร์ดแวร์ โดยเครื่องมือสังเคราะห์วงจรจะตรวจสอบว่าวงจรที่ผู้ออกแบบฮาร์ดแวร์ได้ออกแบบไว้นั้นเป็นวงจรตรรกะเชิงผสมซึ่งอยู่ในขอบเขตของวิทยานิพนธ์หรือวงจรเชิงลำดับซึ่งไม่ได้อยู่ในขอบเขตของวิทยานิพนธ์นี้ ถ้าวงจรดังกล่าวมีลักษณะของวงจรเชิงลำดับแล้วเครื่องมือจำลองการทำงานจะแสดงหน้าต่างโต้ตอบเพื่อแจ้งให้ผู้ออกแบบฮาร์ดแวร์ทราบว่าวงจรที่ได้ออกแบบไว้นั้นมิใช่วงจรตรรกะเชิงผสม ดังแสดงในรูปที่ ก.33



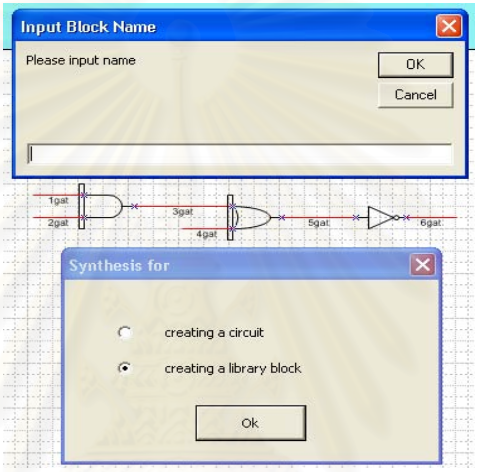
รูปที่ ก.33 หน้าต่างโต้ตอบเพื่อแจ้งว่าวงจรที่ได้ออกแบบไว้มิใช่วงจรตรรกะเชิงผสม

แต่ถ้าวงจรที่ผู้ออกแบบฮาร์ดแวร์ได้ออกแบบไว้ อยู่ในรูปแบบของวงจรตรรกะเชิงผสมแล้ว เครื่องมือสังเคราะห์วงจรก็จะสังเคราะห์วงจร และผลลัพธ์ที่ได้จากการสังเคราะห์วงจรก็คือ

ไฟล์ชนิดข้อความที่มีชื่อว่า netlist.txt ซึ่งเป็นไฟล์ที่จะเก็บภาษาเนทลิสต์จากการสังเคราะห์วงจร เพื่อใช้สำหรับจำลองการทำงานของวงจรต่อไป

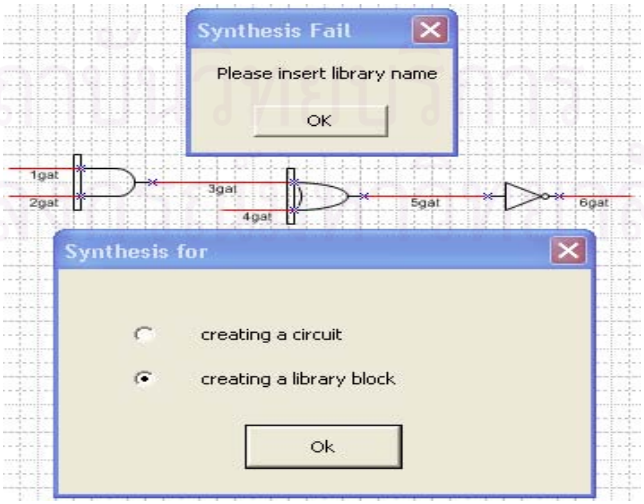
ก.4.2 การสังเคราะห์วงจรเพื่อใช้สำหรับสร้างเป็นวงจรบล็อก

เมื่อผู้ออกแบบฮาร์ดแวร์เลือก Creating a library block จากหน้าต่างโต้ตอบเพื่อเลือกวัตถุประสงค์ของการสังเคราะห์วงจรแล้ว เครื่องมือสังเคราะห์วงจรจะเริ่มทำงาน โดยการแสดงหน้าต่างโต้ตอบเพื่อให้ผู้ออกแบบฮาร์ดแวร์ได้กำหนดชื่อของวงจรบล็อกที่ได้ออกแบบไว้ ดังแสดงในรูปที่ ก.34



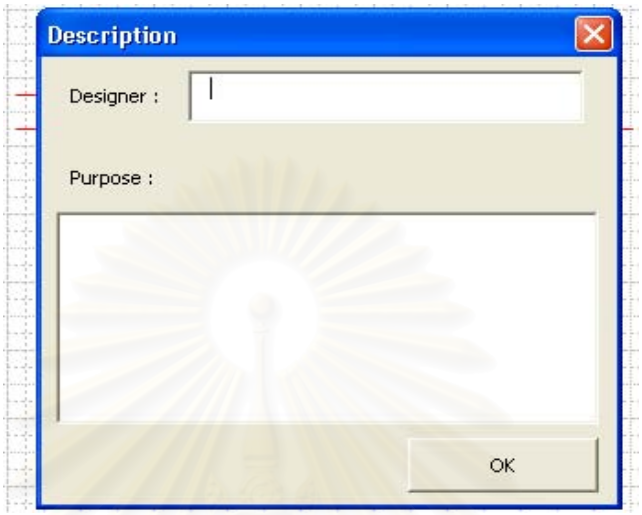
รูปที่ ก.34 หน้าต่างโต้ตอบเพื่อกำหนดชื่อให้กับวงจรบล็อก

ถ้าผู้ออกแบบฮาร์ดแวร์ไม่ได้กำหนดชื่อให้กับวงจรบล็อก โดยผู้ออกแบบฮาร์ดแวร์กดปุ่ม OK เลยจะปรากฏหน้าต่างโต้ตอบสำหรับเตือนผู้ออกแบบฮาร์ดแวร์ให้กำหนดชื่อของวงจรบล็อกที่ได้ออกแบบไว้ ดังแสดงในรูปที่ ก.35



รูปที่ ก.35 หน้าต่างโต้ตอบเพื่อให้ผู้ออกแบบฮาร์ดแวร์กำหนดชื่อของวงจรบล็อก

เมื่อผู้ออกแบบฮาร์ดแวร์กำหนดชื่อของวงจรบล็อกเป็นที่เรียบร้อย หน้าต่างตอบโต้เพื่อให้ผู้ออกแบบฮาร์ดแวร์กำหนดชื่อของผู้ออกแบบฮาร์ดแวร์และวัตถุประสงค์ของวงจรจะแสดงขึ้น ดังแสดงในรูปที่ ก.36

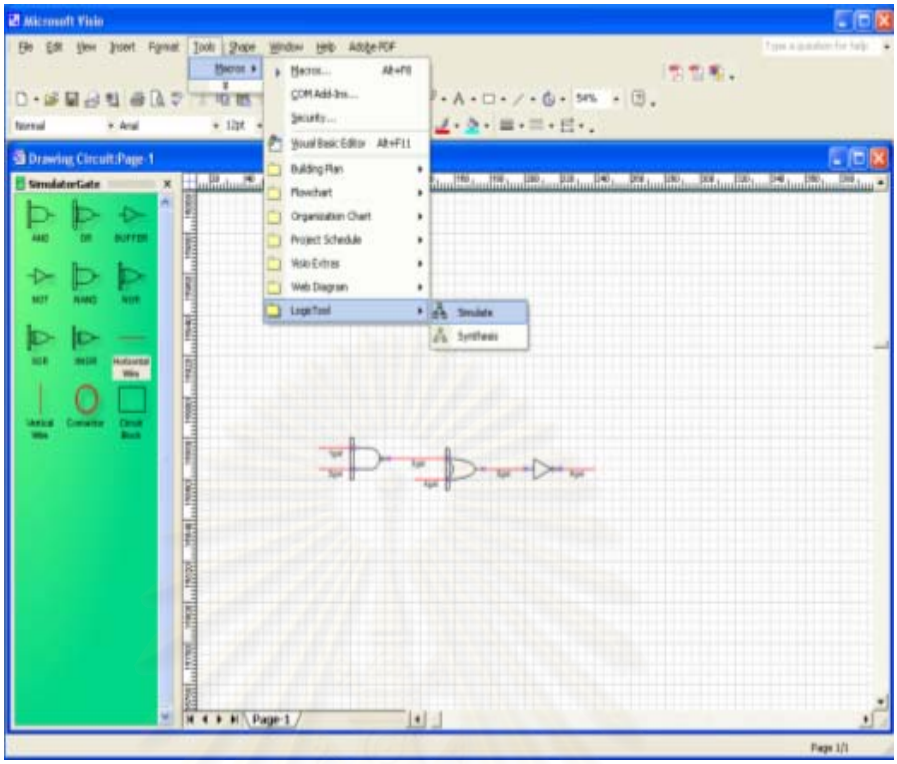


รูปที่ ก.36 หน้าต่างโต้ตอบเพื่อกำหนดชื่อผู้ออกแบบฮาร์ดแวร์และวัตถุประสงค์การออกแบบ

เครื่องมือสังเคราะห์วงจรจะสังเคราะห์วงจรแล้วให้ผลลัพธ์ในรูปแบบของไฟล์ชนิดข้อความจำนวนสองไฟล์ คือ 1) ไฟล์ข้อความที่มีชื่อตามที่ผู้ออกแบบฮาร์ดแวร์กำหนด ซึ่งเป็นไฟล์ที่จะเก็บภาษาเนทลิสต์จากการสังเคราะห์วงจร เพื่อใช้สำหรับออกแบบวงจร และ 2) ไฟล์ข้อความที่มีชื่อตามที่ผู้ออกแบบฮาร์ดแวร์กำหนด + Detail ซึ่งเป็นไฟล์ที่ใช้สำหรับเก็บรายชื่อของอินพุตหลักและเอาต์พุตหลัก เพื่อใช้ในการแสดงทางหน้าต่างโต้ตอบเมื่อวงจรบล็อกถูกเรียกใช้ในการออกแบบวงจร

ก.5 การใช้งานเครื่องมือจำลองการทำงาน

เครื่องมือจำลองการทำงานที่ได้พัฒนาขึ้น จะสามารถเรียกทำงานได้ก็ต่อเมื่อผู้ออกแบบฮาร์ดแวร์ได้ออกแบบวงจรตรรกะเชิงผสมเสร็จและสังเคราะห์วงจรเป็นที่เรียบร้อยแล้ว ซึ่งผู้ออกแบบฮาร์ดแวร์สามารถเรียกใช้เครื่องมือจำลองการทำงาน โดยเรียก Tools -> Macros -> Logic Tool -> Simulate ที่ตำแหน่งเมนูบาร์ ดังแสดงในรูปที่ ก.37 เมื่อผู้ออกแบบฮาร์ดแวร์เรียกเครื่องมือจำลองการทำงานเพื่อใช้งาน เครื่องมือจำลองการทำงานจะเริ่มต้นทำงานโดยการทำงานจะแบ่งเป็น 2 แบบตามลักษณะของวงจรตรรกะเชิงผสม คือ 1) วงจรตรรกะเชิงผสมที่มีเฉพาะเกทพื้นฐาน และ 2) วงจรตรรกะเชิงผสมที่ประกอบไปด้วยเกทพื้นฐานและวงจรบล็อก



รูปที่ ก.37 การเรียกใช้เครื่องมือจำลองการทำงาน

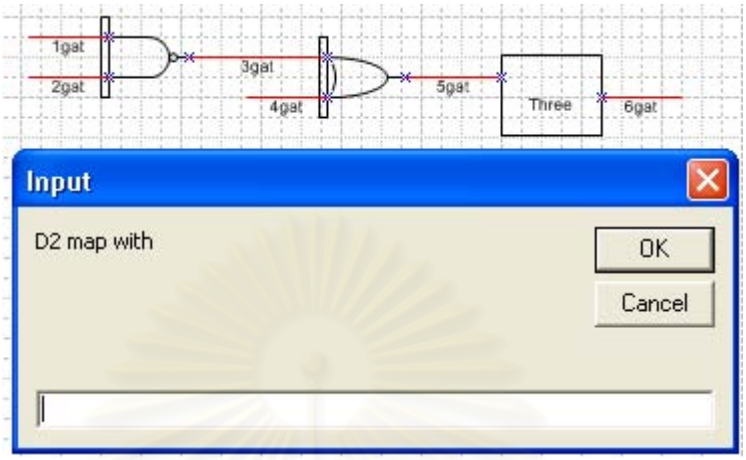
ก.5.1 วงจรตรรกะเชิงผสมที่มีเฉพาะเกทพื้นฐาน

เมื่อเครื่องมือจำลองการทำงานได้จำลองการทำงานวงจรตรรกะเชิงผสมที่มีเฉพาะเกทพื้นฐาน เครื่องมือจะพิจารณาจำนวนของอินพุตหลักว่ามีจำนวนมากกว่าเจ็ดอินพุตหลักหรือไม่ ถ้าจำนวนอินพุตหลักมีจำนวนน้อยกว่าเจ็ดอินพุตหลัก เครื่องมือจำลองการทำงานจะจำลองการทำงานโดยกำหนดค่าเริ่มต้นให้กับอินพุตหลักตามกรณีของเลขฐานสองจนครบทุกกรณี แล้วก็คำนวณหาค่าความจริงของเอาต์พุตหลักจากวงจรที่ได้ออกแบบไว้ แต่ถ้าจำนวนอินพุตหลักมีจำนวนมากกว่าเจ็ดอินพุตหลัก เครื่องมือจำลองการทำงานจะสุ่มค่าความจริงเริ่มต้นให้กับอินพุตหลักนั้น จากนั้นก็จะคำนวณหาค่าความจริงของเอาต์พุตหลัก หลังจากที่เครื่องมือจำลองการทำงานได้ทำงานเสร็จเรียบร้อยแล้ว ผลของการจำลองการทำงานจะอยู่ในรูปแบบของไฟล์ชนิดข้อความชื่อ simulateResult.txt ซึ่งเป็นไฟล์ที่แสดงถึงค่าของสายสัญญาณทุกสายสัญญาณที่ได้ออกแบบขึ้นในวงจรเมื่อมีการคำนวณค่าของเกทแต่ละเกท

ก.5.2 วงจรตรรกะเชิงผสมที่ประกอบไปด้วยเกทพื้นฐานและวงจรบล็อก

เมื่อเครื่องมือจำลองการทำงานได้จำลองการทำงานวงจรตรรกะเชิงผสมที่ประกอบไปด้วยเกทพื้นฐานและวงจรบล็อก เมื่อเครื่องมือได้จำลองการทำงานในส่วนของวงจรบล็อกแต่ละบล็อก

จะปรากฏหน้าต่างโต้ตอบขึ้นเพื่อให้ผู้ออกแบบฮาร์ดแวร์ได้กำหนดชื่อของสายสัญญาณที่ได้ออกแบบไว้กับอินพุตหลักและเอาต์พุตหลักของวงจรบล็อก ดังแสดงในรูปที่ ก.38



รูปที่ ก.38 หน้าต่างโต้ตอบเพื่อกำหนดชื่อสายสัญญาณอินพุตหลักและเอาต์พุตหลัก

เมื่อผู้ออกแบบฮาร์ดแวร์กำหนดสายสัญญาณให้กับอินพุตและเอาต์พุตหลักของวงจรบล็อกเป็นที่เรียบร้อยแล้ว เครื่องมือจำลองการทำงานก็จะจำลองการทำงานจนกระทั่งเสร็จ เครื่องมือจำลองการทำงานจะให้ผลลัพธ์ในรูปแบบของไฟล์ชนิดข้อความชื่อ simulateResult.txt ซึ่งเป็นไฟล์ที่แสดงถึงค่าของสายสัญญาณทุกสายสัญญาณที่ได้ออกแบบขึ้นในวงจรเมื่อมีการคำนวณค่าของเกทแต่ละเกท

ภาคผนวก ข
ผลงานตีพิมพ์

งานประชุมวิชาการวิทยาศาสตร์และวิศวกรรมศาสตร์คอมพิวเตอร์นานาชาติ (The 2003 International MultiConference in Computer Science and Engineering, SERP'03) ระหว่างวันที่ 23 - 26 มิถุนายน 2546 เมืองลาสเวกัส รัฐเนวาดา ประเทศสหรัฐอเมริกา ในบทความเรื่อง An Application of Object Oriented Paradigm (OOP) to Combination Logic Design

An Application of Object Oriented Paradigm (OOP) to Combination Logic Design

S. Chaiworawitgul, P. Pitsatorn, and B. Sowanwanichakul
Department of Computer Engineering
Faculty of Engineering, Chulalongkorn University
Bangkok, 10330, Thailand

Abstract

The libraries that must be used directly and cannot be modified, or while the little function or property are changed mostly causes redesign of a circuit. Those are the problems that all hardware designers are facing currently. This research proposes a new tool to help hardware designers in the process of hardware design, specifically to the combination logic design. This tool acquires the properties of object orientation (OO) concepts namely inheritance and reusability. The proposed tool encompasses a hardware design drawing, a synthesizer, and a simulator with a friendly graphical user interface. Hardware designers use this tool to draw the design of the hardware using drag and drop utilities help ease the process of design and hence advantageous for hardware designers. The tool will then synthesize, and simulate the complete systems exploiting OO connotation. The output of the system (the combination logic circuit) is verified by comparing the result with the standard circuit by International Symposium Circuit and Systems 1985 (ISCAS85). Finally, this research certainly promotes the complement of utilizing the useful concept of software in the world of hardware.

Keywords: Object oriented concept, Inheritance, Reusability, and Combination logic design

1. Introduction

So far, Hardware design process is less flexible than software design process such as when microprocessor, responsible to compute the processes, is designed, designer must specify the constant input and output bits of microprocessor. If designer want to design a new microprocessor that has similar function to the one that is designed before but has a number of differences in input or output, she will have to redesign it from scratch, Currently, both very high speed integrated circuit hardware description language (VHDL) and Verilog, hardware design language, have a library or intellectual property (IP) that appears to be reusable. Nonetheless, in practice, those libraries or IPs are used directly rather than reused. Designers can't override or add properties and behaviors of the libraries or IPs. Therefore it is not flexible for designers and causes them suffering from redundant drawings and tedious calculation of rework. The software design process is a counterpart.

The software design process changed to a new paradigm which is Object Oriented (OO). OO helps design process more flexible than traditional design with the 3 properties encapsulation, inheritance and polymorphism. A designer needs not to redesign resemble objects repetitively but rather inherits and extends it accordingly to a new requirements. This is a example of reusability that uses the concept of inheritance properties.

From the above paragraph, the problem of hardware design process shares the same characteristics of software designing and hence could be solved by OO methodology. The mechanisms, encapsulation and inheritance, are directly corresponding to the problem.

The objective of the paper is to propose a new tool to help hardware designers in the process of hardware design, specifically to the combination logic design in the gate level. The proposed solution employs distinctive characteristics of OO mentioned above especially inheritance and reusability. These two properties would significantly reduce complexities and redundancies in the hardware design process as to be pinpointed later in this paper.

Section 2 introduces previous works which how OO is applied to hardware design. Section 3 introduces how OO concepts are applied in this tool. Section 4 presents the research flow and all tools which are created in this research. Section 5 introduces the verification process of this tool. Section 6 shows an example of a hardware design by object oriented design approach with this tool, and the last section is a conclusion for this research.

2. Previous work

Because OO concept has been adopted successfully in the field of software, it is also widely applied to various fields of computer. Applied OO to digital hardware design is a promising field that is concentrated by this paper. First, The researches which extended hardware description language (HDL) with object oriented concepts [1][2][3][4] and reusability for digital design which is composed of design for reuse and take the reuse component into design [5].

All of researches are successful to improve the performance of hardware process. They show that OO is powerful to apply in hardware process however, the first group researches are concentrated on hardware description language that are not concentrated on graphic design tool and the second group proposes how to reuse a component in design.

This paper proposes a new graphic design tool, applied OO concepts, to improve the convenient of design over text base. Beside, this tool supports reusability concept to enhance the efficiency of design process.

3. The Object oriented design approach

This research applies object oriented concepts into combination logic design. For a proof of concept, the research mainly focuses on gate level because of its simplicity. However, note that the application can be extended to other subfields. The object oriented concepts used in this research are mainly made up of inheritance and reusability concepts.

3.1. Inheritance

In Object Oriented Paradigm (OOP), all of classes are basically composed of properties or behaviors or both of these. In OOP, inheritance means a new type of object can be created and inherits all of the characteristics of an existing object (referred to as base type).

When applied to gate level design, Gate and wire are conceptualized generically as base classes like those in OOP. The properties of Gate class are bit values of signal, input and output of each gate, whereas the behavior of Gate class is to compute output signal from input signal. There are 7 basic gates used in this research i.e., NOT, AND, OR, NAND, NOR, XOR, and XNOR gates. These are gate types inherited from the base class and therefore share the same properties and behaviors as those of their base class. However, There exist differences in behavior details among them, for instance, AND yields output 0 when at least one of the input is 0 whereas OR returns 1 if at least one of the input is 1. NOT assign an opposite value to the input signal as an output. These characteristics can be captured in the solution by extending the base class. Moreover, among gates that require more than one input can be inherited and defined as different types. For example, AndGateTwo class represents AND gate that comprises two inputs. The same logic applies to AndGateThree, AndGateFour, etc.

Moreover, Wire class consists of property, that is bit Wire class. When a wire is connected to the existed wire, the new wire object call a method to set a value as the value signal, and behavior that sets, gets and shows value of signal. All of wires that are used in design are object of value of the existed wire object. The inheritance property can be applied to parallel data transfer. For example, WireTwo class represents wire that comprises two bit. The analogous logic applies to WireThree, WireFour, etc.

3.2. Reusability

Currently, both very high speed integrated circuit hardware description language (VHDL) and Verilog language have a library or intellectual property (IP) that seems to be reusability concept but library or IP is used directly by the designers. Designers can't override or add properties and behaviors of library or IP, so It is not flexible for designers. This proposed approach is more flexible than before because all of the designs that are completed by designer can be stored as object classes which contain properties and behaviors. The designers can inherit and extends the properties and the functionalities of the existing objects, For example, FullAdderOneBit class is a class that is a completed design for calculating the result of two numbers of 1 bit and carry in 1 bit that has 2 output values, carry out and sum value. These bits and full adders behaviors are named as properties and summation. When designers want to design 2 bit full adder, they can reuse FullAdderOneBit class as a base class and add property that necessary for a new design or override behavior for a new design. Besides, designers can bring a design that is completed to use as a part of a new design directly by not to modify it.

4. OOP-Based designed tool and its components – combination logic design with object oriented

Based on the concept of OOP mentioned in the previous section, the tool has been designed to facilitate the hardware design process. It comprises 3 main components, i.e. Hardware Design Drawing, Synthesizer, and Simulators. Description of each of them will be described in the following subsections.

Hardware design drawing is a component that is designed first. OOP concept is applied to this component. When it completed, synthesizer, used to generated netlist from hardware design drawing, is created. After two components are finished, They are verified by using a standard combination logic design ISCAS85 benchmark.

Activity diagram of working processes is shown in Figure 1.

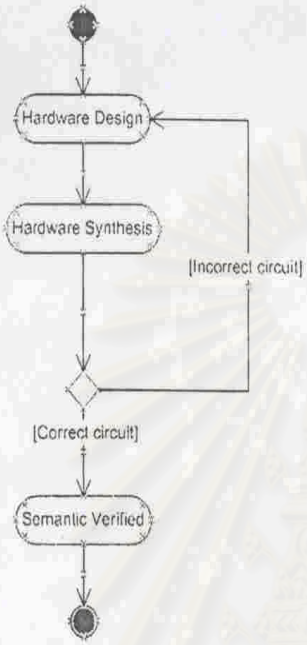


Figure 1 : An activity diagram of working processes.

4.1. A hardware design drawing

This tool is developed in Microsoft Visio. The tool supports 7 basic gates which are NOT, AND, OR, XOR, XNOR, NAND, and NOR gates and supports wires which is used to connect all of signals. Every type of gate supports inheritance concept. Except NOT, XOR, XNOR gates, every type of gate has a rectangle that is adjacent to a gate in Figure 2. This rectangle represents an interface for input wires. Its size is varied with the number of the input wires. Designers can drag and drop an appropriate gate type and then wire it appropriately. The tool will detect the number of the input wires and replace the label accordingly. For example, if the gate is of type 'AND' and the number of connected input wires is 3, the label of that gate will appears as 'AND-3' automatically. Note that although the number of input may vary among the same gate type, their core functionalities are the same. This is captured by OOP introduced in our design framework. The previous designs can be collected in a library system. All of designs in the library not only can be reused directly but also can reused for a part of another design by hardware designers. A point that a new tool is better than recent commercial tools is that they consist of all kinds of gate. In most commercial tools, there are several types of gates predefined according to the number of input although they share the same functionality (recall an example of AND gates). There are many symbol of gate that are redundant, not flexible, and take many resources of program. Figure 3 shows an example of comparison of objects in this tool and those in LogicWorks Design tool (Capilano)

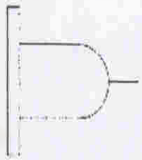


Figure 2 : An example of gate in the tool.

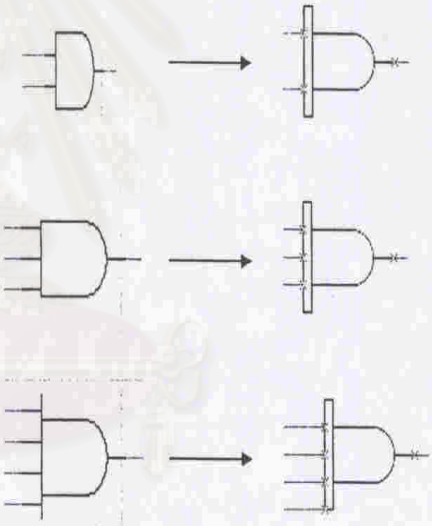


Figure 3 : An example of a different input interface between Logicworks Design Tool (LHS.) and this tool (RHS.)

Figure 4 shows an example of this tool which is used to draw one bit full adder. The boolean equation of full adder 1 bit is

SUM = (/A • /B • C) + (A • /B • /C) + (A • B • C)
COUT = (A • B) + (A • C) + (B • C)

/, •, + represents NOT, AND, OR correspondingly.

Besides, when a designer finishes a complete circuit, the tool supports saving its and it can be reused like a black block to design other circuits like Figure 5. A designer may not necessarily know how to implement a particular block but he need only know its functions and

interfaces. This concept leverages the space design problem and supports a system design that will be complex for designer.

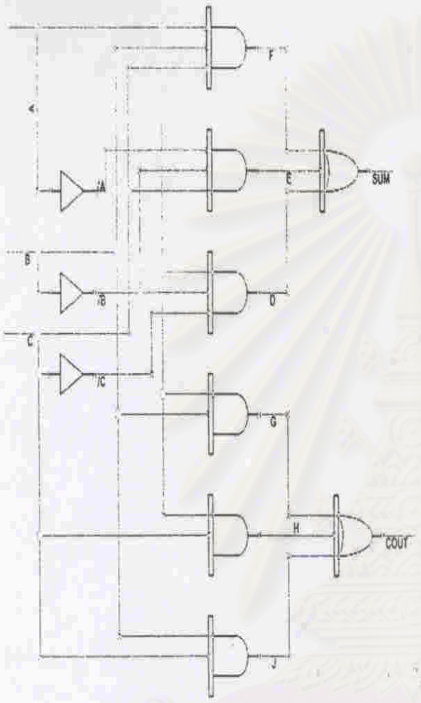


Figure 4 : One full adder bit, designed by object oriented design approach on this tool.

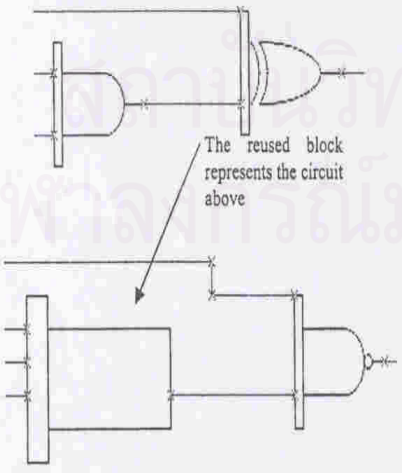


Figure 5 : An example of reusability which is supported in this tool.

4.2. Synthesizer

Synthesizer is used to synthesize a combination logic design with OO design approach from a hardware design drawing. A graphic design from hardware design drawing tool is then converted to a standard netlist language of ISCAS85 by the synthesizer. It uses graph theorem to find the relation of signals that are input and output of each gate from each wire. During synthesizing, the synthesizer verifies syntactic of a design. If the syntactic is false, the synthesizer will not generate a netlist and will notify designers to check all errors. Figure 6. shows the example of false syntactic and screenshot of this notification and Figure 7 shows the example netlist of design in Figure 4

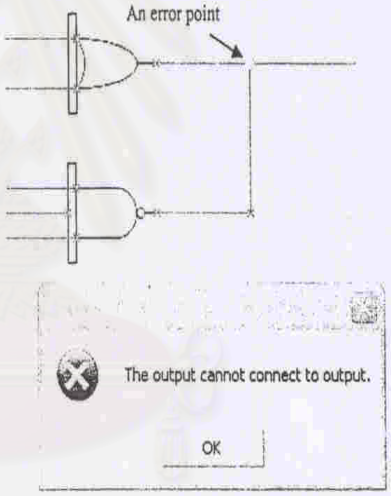


Figure : 6 An example of false syntactic.

```
INPUT(A)
INPUT(B)
INPUT(C)
OUTPUT(SUM)
OUTPUT(COUT)
/A = not(A)
/B = not(B)
/C = not(C)
D = and(A, /B, /C)
E = and(/A, /B, C)
F = and(A, /B, C)
G = and(A, B)
H = and(A, C)
J = and(B, C)
SUM = or(D, E, F)
COUT = or(G, H, J)
END
```

Figure 7 : The netlist language of one bit full adder is generated by synthesizer.

4.3. Simulator

The last component of this tool is a simulator that use to simulate and to show the result of design in graphic format. The standard netlist language of ISCAS85, resulted from synthesizer, is simulated by simulator which represents the result as a wave form. Figure 8 shows an example of wave form that simulated from the netlist language in figure 7.

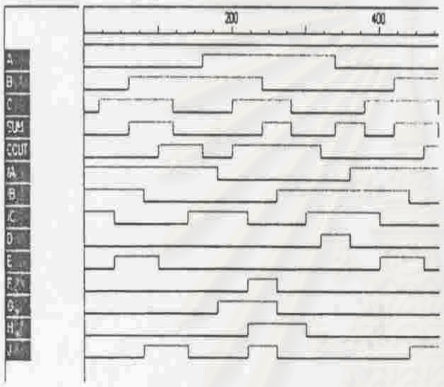


Figure 8 : A wave form of one bit full adder generated from simulator in this tool.

5. Tool applicability verification

The combination logic design with object oriented design approach is analogized by comparing the netlist language that generated from synthesizer with a standard combination logic design ISCAS85. The processes to analogize are

5.1. A standard combination logic design ISCAS85

A standard combination logic design ISCAS85, a netlist language, is made up of 11 designs which are C17.BENCH, C432.BENCH, C439.BENCH, C880.BENCH, C1355.BENCH, C1908.BENCH, C2670.BENCH, C3540.BENCH, C5315.BENCH, C6288.BENCH, and C7552.BENCH. All of these designs must be verified with the netlist languages that is generated by the current design approach. Figure 9 shows the C17.BENCH that is a one of standard combination logic design ISCAS85.

5.2. Comparison results

The semantic of 11 netlist languages of a standard combination logic design ISCAS85 that are generated by synthesizer which have input as design drawing with OO design approach are compared to the original netlist language of combination logic design ISCAS85. If the result from comparing all of 11 designs is precise, it is a sufficient evidence of credibility of the current design with OOP approach.

```
INPUT(1gat)
INPUT(2gat)
INPUT(3gat)
INPUT(6gat)
INPUT(7gat)
OUTPUT(22gat)
OUTPUT(23gat)
10gat = nand(1gat, 3gat)
11gat = nand(3gat, 6gat)
16gat = nand(2gat, 11gat)
19gat = nand(11gat, 7gat)
22gat = nand(10gat, 16gat)
23gat = nand(16gat, 19gat)
END
```

Figure 9 : An example of a standard netlist C17.BENCH design

6. An example

This section represents an example of designing by OO design approach by this tool Figure 10 shows an example of drawing C17.BENCH that has a boolean expression

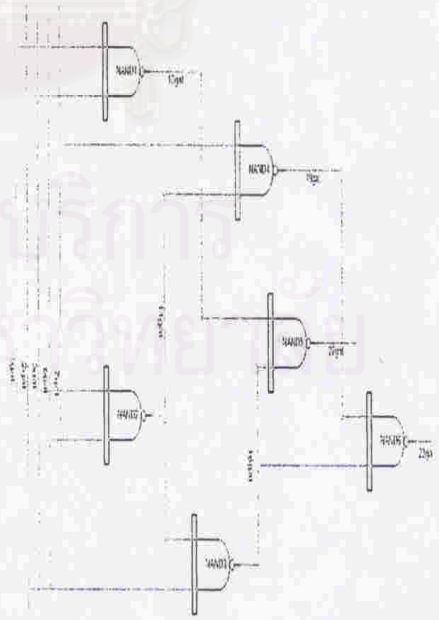


Figure 10 : An example of drawing C17.BENCH by this tool.

22gat = /[/(1gat • 3gat) • /{2gat • /(3gat • 6gat) }]
23gat = {/(2gat • /(3gat • 6gat)) • /(/(3gat • 6gat) • 7gat) } by this tool.

The design must be verified the syntactic by synthesizer. It generates netlist language, likes Figure 9.

Design's netlist language is used to generated wave form by simulator. Figure 11 shows the wave form of C17.BENCH.

From the comparison, a semantic of netlist that is generated from synthesizer is match to the semantic of C17.BENCH. It shows that a tool can be work properly.

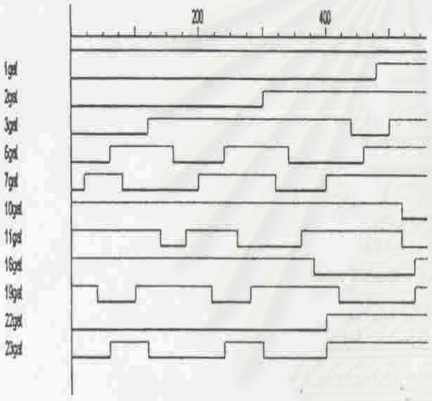


Figure 11 : An example wave form of C17.BENCH design generated from simulator of this tools.

7. Conclusion

This paper has proposed a new tool which applied inheritance mechanism, one of the most powerful feature in OO, to design a combination logic in gate level. The tool includes a hardware design drawing, a synthesizer, and a simulator. The results from OO design approach is verified by comparing the semantic of netlist language of all ISCAS85 designs which are generated by the OOP-based tool with the original netlist of combination logic design ISCAS85. Verification result shows that the tool can replicate semantic in ISCAS85 and therefore, the tool credibility is certified. Therefore, it has been shown that the concept of OO can be applied to the hardware design problem which leads to reduction of complexities and redundancies in the typical hardware design process.

8. References

[1] MAGINOT S., "Evaluation criteria of HDLs : VHDL compared to Verilog, UDL/I & M.", *Proc. ACM SIGDA Design Automation Conference*, ACM Press., 1992, 746-751.

[2] SCHUMACHER G. and NEBEL W., "Inheritance concept for signals in object-oriented extensions to VHDL.", *Proc. ACM SIGDA European Design Automation Conference with EURO-VHDL'95 on EURO-DAC*, ACM Press., 1995, 428-435.

[3] SMITH J., "VHDL & Verilog compared & contrasted – plus modeled example written in VHDL, Verilog and C.", *Proc. ACM SIGDA Design Automation Conference*, ACM Press., 1996, 771-776.

[4] ASHENDEN J. and WILSEY A., "Considerations on object-oriented extensions to VHDL.", *Proc. VIUF Conference*, VIUF Press., 1997, 109-118.

[5] JACOME F., and PEIXOTO P., "A survey of digital design reuse.", *Proc. IEEE Design & Test of Computer*, IEEE Press., 2001, 98-107.

ประวัติผู้เขียนวิทยานิพนธ์

นายศรัณย์ ชัยวรวิทย์กุล เกิดเมื่อวันที่ 17 มกราคม พ.ศ. 2523 ที่จังหวัดกรุงเทพมหานคร สำเร็จการศึกษาหลักสูตรวิศวกรรมศาสตรบัณฑิต (วศ.บ.) สาขาวิศวกรรมคอมพิวเตอร์ จากภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย เมื่อปีการศึกษา 2543 และเข้าศึกษาต่อหลักสูตรวิศวกรรมศาสตรมหาบัณฑิต (วศ.ม.) สาขาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์ จุฬาลงกรณ์มหาวิทยาลัย เมื่อปีการศึกษา 2544 ขณะศึกษาได้มีโอกาสไปเสนอผลงานเรื่อง An Application of Object Oriented Paradigm (OOP) to Combination Logic Design ในงานประชุมวิชาการวิทยาศาสตร์และวิศวกรรมศาสตร์คอมพิวเตอร์นานาชาติ (The 2003 International MultiConference in Computer Science and Engineering, SERP'03) เมืองลาสเวกัส รัฐเนวาดา ประเทศสหรัฐอเมริกา